Research on Ground Plane Detection and Indoor Scene Labeling with Depth Information

September 2017

Graduate School of Systems Engineering

Wakayama University

Yankun Lang

Abstract

This thesis describes three new algorithms for solving the problems of Ground Plane Detection (GPD) and 3D Indoor Scene Labeling (3D-ISL). Chapter 1 gives a brief discussion about background, including the necessity, the application and the related works for solving the problems, as well as the advantages and limitations of existing methods and algorithms.

The second chapter describes a height distribution based algorithm for detecting either a single or multiple ground planes from a single 3D indoor scene captured by a RGB-D camera. One of the reasons that makes the detection of multiple planes from a cluttered scene difficult is the high dimension of the parameter space. In general, conventional methods use statistical approach in three dimensional parameter space to detect planes. However, the sparse distribution in such space is difficult to converge thus fails to give stable results. In this research, the dimension of the parameter space is reduced from three to one by making use of the predictable camera pose. The distribution obtained with statistical processing of the input data will become well shaped so that it makes the detection of ground planes much easier and stable.

This algorithm consists of two phases: camera pose estimation and ground plane detection. Starting with a predicted camera pose, both the ground planes and the accurate camera pose are refined through a certain number of iterations. The detection of the ground planes are carried out by the following steps:

1. Computing the distribution in the one dimensional parameter space with a known (or predicted) camera tilt angle.
2. Utilizing this distribution to detect the ground planes and estimate degree of their convergence.

3. Finding the camera tilt angle that gives the best convergence degree.
4. Repeating step.1-3 until both the detected planes and camera tilt angle are convergent.

In addition to GPD, the task of 3D Indoor Scene Labeling (3D-ISL) is also studied in this thesis. The correctness of labeling depends on two factors: 1) the distinctiveness of the extracted features and, 2) the correctness of the label relevance. Chapter 3 and Chapter 4 focus on these two factors.

In the third chapter, an algorithm of labeling a 3D indoor scene captured from a fixed RGB-D sensor is proposed. The algorithm proposed in chapter 2 is utilized here to improve the accuracy of 3D-ISL. A discriminative feature describing the spatial distribution characteristics of 3D objects is generated. Meanwhile, label relevance in the 3D indoor scene are modeled. The labeling problem is solved by using the graphical model.

In order to achieve an accurate result with low computational cost, the following ideas are used in this algorithm:

1. Using Eigenvector decomposition and sub-space combination for making a strong spatial feature.
2. Designing a 6-connected pair-wise model to represent the label relevance.
3. Defining each part of the cost function corresponding to the feature vector and the label relevance respectively.

In the fourth chapter, another algorithm of 3D-ISL is proposed. In order to make the labeling result robust against the camera rotation, a high dimensional feature vector is generated by combining several single rotation invariance features together. Furthermore, a method of learning the label relevance is also proposed to improve the accuracy. At last, a method of detecting person object from the labeling result is proposed.

Numerous experiments have been carried out to evaluate the performance of our algorithms of Ground plane detection and 3D indoor scene labeling. The advantages and disadvantages of our works are discussed in the last chapter.

概要

本論文では、地面検出と 3D 屋内シーンのラベリングの問題を解決するために 3 つのアルゴリズムを提案する。第 1 章では、研究目的・背景、必要性、応用、従来手法の利点と限界について述べる。

第 2 章では、RGB-D カメラで撮影された 3D 屋内シーンから単一および複数の地面を検出するために、高さヒストグラムベースのアルゴリズムを提案する。複雑なシーンから複数の平面を検出することを困難にする理由の 1 つは、パラメータ空間の高次元である。従来手法では、3 次元パラメータ空間における統計的アプローチを用いて平面を検出している。しかし、このような 3 次元パラメータ空間における疎な分布は収束しにくく、安定した結果が得られない。本論文では、予測可能なカメラ姿勢を利用して 1 次元パラメータ空間で地面検出を行うアルゴリズムを提案する。提案アルゴリズムは 1 次元パラメータ空間を用いる方法であるため、容易であり、また地面の位置は、シャープな分布として得られるため、安定した検出が可能となる。

このアルゴリズムには、カメラのチルド角度の推定と地面検出の 2 つの処理を含まれる。予測されたカメラ姿勢から始め、繰り返し処理を行うことにより，一枚の深度画像からカメラのチルト角度と，3 次元空間内の地面を含む複数の平面を同時に検出することが可能になる。地面の検出は、以下の手順で行う。

1. 既知の（または予測された）カメラのチルト角を用いて 1 次元パラメータ空間における高さ分布を計算する。
2. 高さ分布を利用してグランドプレーンを検出し、それらの収束の度合いを推定する。
3. より良い度合いを得られるカメラのチルト角を探索する。
4. 地面の検出とカメラチルト角の推定が収束するまで手順 1) から 3) まで繰り返し処理を行う。

3D 室内シーンのラベリングの正確さは、抽出された特徴の識別性と、ラベル関連性の正確性に依存する。第 3 章と第 4 章では、この 2 つの要素について提案する。

第 3 章では、地面検出の結果を活用し、固定 RGB-D センサで撮影された 3D 屋内シーンのラベリングのアルゴリズムを提案する。3D オブジェクトの空間分布を記述する識別特徴が生成する。同時に、3D 屋内シーンにおけるラベル関連性がモデル化する。3D 屋内シーンのラベリングの問題は、グラフィカルモデルを使用して解決する。計算コストを削減し、精度を向上するために、提案手法では以下のアイデアを含む。

1. 強い空間的特徴を作るために固有値分解と部分空間の構築を用いる。
2. ラベル関連性を表現するために、6 連結モデルを設計する。
3. 構築する特徴ベクトルと設計するラベル関連性に対応するコスト関数をそれぞれ定義する。

第 4 章では、姿勢が任意に変化する RGB-D センサで撮影された動画から 3D 室内シーンのラベリングアルゴリズムを提案する。提案アルゴリズムでは、いくつかの単一の回転不変特徴を組み合わせることによって高次元特徴ベクトルを生成する。さらに、精度を向上させるためにラベル関連性を学習する方法も提案する。そして，オブジェクトとしてラベリングされたものから人物の検出を行う。

従来手法との比較実験より，提案特徴と提案手法の有効性をそれぞれ確認できた。

Declaration

Some parts of the work presented in this thesis have been published in the following articles:

Journal Paper

1 **Yankun Lang**, Haiyuan Wu, Toshiyuki Amano, Qian Chen, "3D Single/Multiple Ground Planes Detection with Camera Angle Estimation",The Journal of the Institute of Image Electronics Engineers of Japan (IIEEJ), IIEEJ Transactions on Image Electronics and Visual Computing, Vol.5, No.1, 2017, Jun.

International Conference With Review (Relates to this thesis)

2 **Yankun Lang**, Haiyuan Wu, Qian Chen, "3D indoor scene labeling and people detection approach based on rotation invariant features", IEEE International Conference on Image Processing (ICIP), 2017 (Accepted)

3 **Yankun Lang**, Haiyuan Wu, Qian Chen, "Spatial distribution feature for 3D indoor scene labelling", 3rd IAPR Asian Conference on Pattern Recognition (ACPR), pp.066-070, 2015.

4 **Yankun Lang**, Haiyuan Wu, Toshiyuki Amano, Qian Chen, "An iterative convergence algorithm for single/multi ground plane detection and angle estimation with RGB-D camera", Image Processing (ICIP), 2015 IEEE International Conference, pp.2895-2899, 2015.

International Conference With Review (Not relates to this thesis)

5 Mineyuki Tsuda, **Yankun Lang**, Haiyuan Wu, "Analysis and Identification of the EEG Signals from Visual Stimulation". Knowledge-Based and Intelligent Information and Engineering Systems 18th Annual Conference (Open access of Procedia Computer Science), KES-2014 Gdynia, Poland, Vol.32, pp.1292-1299, September 2014 Proceedings

6 Yankun Lang, Jiancheng Zou, Haiyuan Wu, Qian Chen, "New Digital Image Compression Framework Based on Compression Sensing and Sparse Representation". 20th Korea-Japan Joint Workshop on Frontiers of Computer Vision, pp.202-207, 2014.

Others

7 Yankun Lang, Haiyuan Wu, Toshiyuki Amano, Qian Chen, "Cross Iterative Method for Angle Estimation and Ground Plane Detection with RGB-D Camera", The 77th National Convention of IPSJ, pp.3ZG-04, 2015.

8 Peng Li, **Yankun Lang**, Qian Chen, Haiyuan Wu, "Person Identification with FREAK", Computer Vision and Image Media, Vol. 2014, Issue 44, pp.1-6, 2014.

9 Peng Li, **Yankun Lang**, Qian Chen, Haiyuan Wu, "Person Identification with FREAK",IEICE Technical Report, MVE, Vol.113, Issue.403, pp.263-268

10 Yankun Lang, Jiancheng Zou, "Digital Image Coding based on Compressed Sensing and Sparse Representation", 9th Annual Conference of China Institute of Communications. 2012 (**Best paper award**)

Acknowledgements

I would like to thank all my colleagues in the Wu Laboratory for many stimulating and profitable discussions. My supervisor Professor Haiyuan Wu has directed me and encouraged me to complete this work and has always been there whenever I have needed her help. Also, she has helped me a lot in life. I would also like to thank Professor Chiaki Sakama, Associate Professor Masato Soga and Associate Professor Qian Chen. The support received from them helped me very much.

Thank Professor Toshiyuki Amano and Associate Professor Kei Iwasaki for their careful reviewing and precious comments on this thesis, because of their help, this thesis has been improved greatly.

Many thanks are due to Associate Professor Masao Ohira, Associate Professor Koichi Ogawara, Ms. Noriko Yamasaki, Mrs. Chunchun Yang as well as other students of Wu Laboratory for their kindly help during my three years study. Thank Doctor Kazumasa Suzuki and Mr. Peng Li for their contributions to this work and the help for my life.

I would like to thank the members of Kainan Eastern Club of the Rotary Yoneyama Memorial Foundation, Inc. for their financial support in my last two years study. Especially thank to my counselor Mr. Munehiro Hanada and Mr. Youichi Sakaguchi. Their encouragements help me to finish my study here.

Finally, I will give my thanks to my family, my parents (my mother Zhaoyang Shen, my father Baohu Lang). Thank their support and encourages in my three years study abroad, because I could not imagine that I can finish my study without their help.

Contents

1

Introduction

1.1 Ground plane detection

Computer vision is a field that deals with the problem how to make computers to understand digital images or videos. In recent years, with the rapid development, it has been widely applied to navigation, surveillance, autonomous cars and automatic robots, etc.. Generally, computer vision consists of several tasks: object identification, object detection, object tracking, motion analysis from extrinsic parameter, scene labeling and classification, etc.. Among them, as the fundamental techniques for many real applications such as automatic robots, surveillance system and tracking system, ground plane detection and scene labeling have received great attentions and are still being developed rapidly. In this chapter, we will introduce the recent studies on these two tasks and discuss the problems of them.

1.1.1 Literature review of the ground plane detection

As a fundamental problem of scene understanding, ground plane detection has been widely used in numerous applications ranging from moving object tracking to automatic identification and intelligence robotics [1, 2, 3, 4, 5]. Ground plane detection aims to detect single or multiple grounds from a complex scene having various appearances of backgrounds, which makes it a challenging work in the computer vision area. Most existing approaches used Euclidean distance as the key to detect the ground plane, where RANSAC has been widely applied due to its practicality and convenience. Recently,

many approaches tend to detect ground plane with the perspective of probability theory [6, 7, 8, 9, 10]. The algorithm proposed in [9] first predicts the depth map of a 2D image through a Markov Random Field (MRF), then divides the image into regions of perceptually similar textures (superpixels). The superpixels that have similar features (color, normal) will be considered as co-planar. At last, the 3D ground plane can be reconstructed by calculating the actual depth of each pixel through a camera calibration.

The most useful approaches of ground detection include: RANSAC, Hough Transform, graphical model, etc.. Meanwhile, based on these approaches, many extensions have been developed in the past few years. According to their characteristics, we classify the approaches of ground detection from 2D image or 3D scene into two main categories: geometric model based and region growing based. Relative methods and studies will be discussed respectively.

Geometric model based

In computer vision, the most widely known methodology of plane extraction based on geometric model is RANSAC, which has been proved to be able to detect planes successfully in 3D scenes. RANSAC is an algorithm initially developed by Fischler and Bolles in [11] that allows the fitting of the model without trying all possibilities. RANSAC is based on the probability to detect a model using the minimal set required to estimate the model. For a given data set whose data elements contain both inliers (ground plane) and outliers (noise), a plane geometric model is initialized by three randomly chosen points according to the fact that a plane can be defined by three points. Then for the rest data set, distance from each point to the model is calculated and, if it is smaller than the predefined threshold, the point will be determined as belonging to the ground. RANSAC uses a score function to find the optimal parameters of the model. Usually, the score is the number of points belonging to the plane.
RANSAC exhibits the following, desirable properties[12]:

1. The algorithm is conceptually simple, which makes it easily extensible and straightforward to implement.

2. It is very general, allowing its application in a wide range of settings.

3. It can robustly deal with data containing more than 50% of outliers.

Apparently, RANSAC is very efficient in detecting large plane in noisy point cloud but becomes very slow to detect small planes. In addition to that, RANSAC will cause failure detection when there are multiple ground planes in the scene since the model is a single plane. Torr et al. proposed MLESAC [13] which is a generalization of the RANSAC estimator. It adopts the same sampling strategy as RANSAC to generate estimated solutions, however, the solution is updated by minimizing the negative log likelihood of the cost rather than just the number of inliers. MLESAC improves the accuracy of ground detection especially in the scene with strong noises (small planes). However, since the model remains single, it is not able to be applied to the detection of multiple planes either.

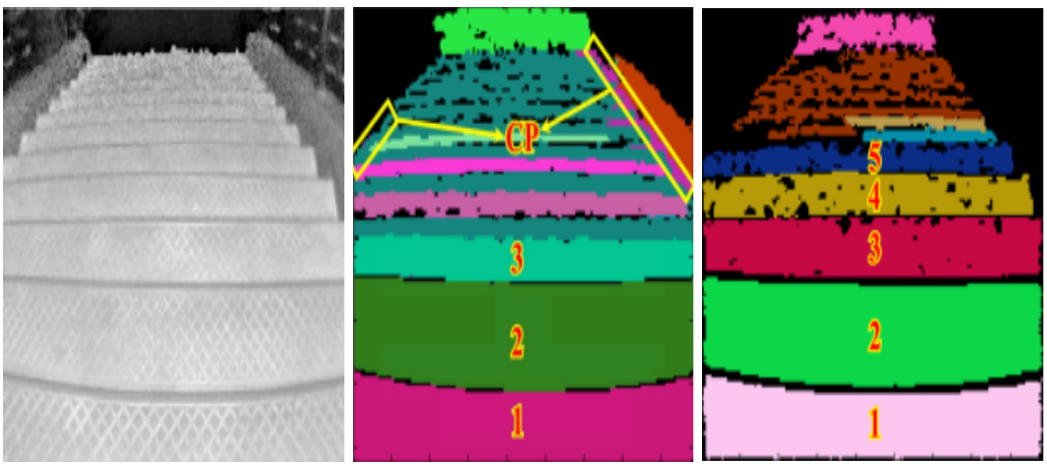
Ruwen Schnabel in [14] proposed a robust and fast method which is an extension to RANSAC for shape extraction including planes. In this research, shape extraction problem was formed as an optimization problem defined by a score function. During each iteration, for each subset sampled by an octree, candidates of all considered shape types are generated, then the best candidate is evaluated through a score function defined by a probability. This method has been proved to be able to detect vast shapes from a noisy scene.

For detecting multiple planes, several extensions to RANSAC have been developed in the recent years. In [15], a sequential RANSAC method extracts multiple planes by using RANSAC to extract the plane containing the most inliers, then removing the inliers from the point set, for the remaining points, repeating the same procedure until all the planes are extracted. This method is simple and easy to implement. However, the removal of inaccurate inliers may adversely impact the subsequent RANSAC processes. Moreover, this method may cause low accuracy when the scene contains several intersecting planes (such as steps), in which case the biggest plane extracted may contain some inliers belonging to other planes.

To overcome the first disadvantage discussed above, Multi-RANSAC method proposed in [16] estimates multiple models in each of the RANSAC iteration. It can detect multiple planes more accurately but with higher computational cost. A clustering algorithm called J-linkage in [17] is used to refine the selected inliers, from which to extract multiple planes. This method has been proven to be more robust against noise.

In order to avoid the second disadvantage brought by connected planes, in Orazio's research[18], CC-RANSAC was proposed that only considers the largest connected components of inliers to evaluate the fitness of a candidate plane. This idea embeds the observation that data points which are the inliers of a correct plane cluster are contiguous in space, whereas a plane straddling across two planar patches typically produces two disconnected sets of inliers. This method overcomes the drawback that RANSAC could not detect multiple planes that are close to each other. However, this method introduces another drawback: it will cause failure when several planes are connected together.

NCC-RANSAC[19] is an extension to CC-RANSAC. Normal coherence is considered in this method, that is, if the angle between the surface normal of the inlier data point and the fitted plane is big, then the inlier data will be discarded, which results in a number of disconnected planar patches. For each of the rest patches, instead of searching the largest patch, a plane is fitted to the patch and the distance from each data of the scene to the plane is calculated, then points with a sufficiently small distance are added to the patch. By repeating these steps, all the planes could be extracted. As a result, NCC-RANSAC method is much more efficient than CC-RANSAC method. The results in [19] are shown in Fig.1.1.



(a) input intensity image (b) results by using CC-RANSAC (c) results by using NCC-RANSAC

Figure 1.1: Comparative results of CC-RANSAC in [18] and [19]

Faisal M. in research[20] proposed a method for ground plane detection in spatio-

temporal volume of images taken from a TOF (Time of Flight) camera. Since in single frame where the number of the points belonging to an obstacle (e.g. a wall) may be more than the ones belonging to the ground, using standard RANSAC may cause a failure. To solve this problem, instead of using a single frame, several continuous frames are used to create a spatio-temporal volume, where a motion constraint was defined that the angular velocity of the camera is parallel to the normal of the ground plane. By using the spatio-temporal features on the ground, spatio-temporal RANSAC was proposed based on this constraint and experimental results showed that using spatio-temporal RANSAC could solve the problem in their work.

In addition to RANSAC, Hough Transform [21] is another typical model-based method used for plane detection. It has been applied to detecting other shapes as well [22, 23, 24, 25, 26]. The basic idea is that each data point casts its vote in the Hough Transform (HT) parameter space. This space is divided into several sub-spaces called accumulator cell, which represents the plane parameters. The accumulator cell with the largest number of votes are identified as the parameters for the plane model. However, since the 3D point cloud contains a large number of points and each of them needs to be transformed, traditional HT usually brings about numerous computational cost. To reduce the computational cost, Kiryati et al. in [27] proposed a probabilistic method called Probabilistic Hough Transform (PHT). In this method, instead of using the whole points, only a subset of points is randomly selected and then transformed into the HT space for voting. By reducing the number of points, the computational cost is reduced drastically. However, the performance of PHT is highly dependent on the size of selected subset.

Based on PHT, Adaptive Probabilistic Hough Transform proposed in [28] developed a novel structure of the accumulator which changes dynamically during the voting phase. After selecting a random set of points and transforming them onto the HT space, during each voting process, the maximal cell is added to a list of potential maximum cells for updating. As updating proceeds, the structure of the accumulator becomes clearer. The updating procedure stops after a certain number of iterations and the result of plane detection will gain a high accuracy. However, since the stopping rule for updating is complex, it is hard to implement.

Matas proposed another HT based method call Progressive Probabilistic Hough Transform (PPHT)[29]. In this method, the planer objects are detected progressively,

meaning that if the highest accumulated cell exceeds the threshold, points corresponding to this cell will be added to the output list and removed from the whole points simultaneously. Same steps repeat for the rest points until a certain condition meets. This method accelerates the speed since not all the points participate in the voting. In Dorit B's research [30], an innovative accumulator was designed to make each cell equally separated. The evaluation results showed that the accuracy could be improved by using this accumulator.

Hough Transform has been widely used for multi-planes detection. Since a plane can be represented by its normal vector and the distance to the origin, the parameter space of the Hough Transform has three dimensions: two for the normal vector and one for the distance. In the case that the voting is performed for every single point in the cloud, each 3D point will form a surface in the parameter space. This is not only computational expensive, but also makes the votes scattered that will make the plane detection difficult. In order to solve this problem, many researches used planar patches instead of a single point for the voting. Randomized Hough Transform (RHT) was proposed in research [31]. In this research for detecting planes, three points are chosen randomly from the input and mapped onto Hough space, represented as one point that denotes the plane spanned by the three points. The corresponding cell in Hough Space calculated by the plane is accumulated. After certain number of iterations, cell with accumulations reaching a certain threshold will be considered as the plane. Then points lying on this plane will be removed from the input. This method has a significant advantage that it is not necessary to perform Hough transformation for all the points thus makes the method efficiency. However, this approach has two obvious disadvantages. The first one is the low accuracy of parameters in the plane patch. This is because a patch only contains a small number of points. This low accuracy makes the votes scattered in a wide range for the patches belonging to the same plane, thus makes the plane detection unstable and inaccurate. The second one is that it cannot guarantee that the points of a patch are selected from the same plane. Selecting points in a small range reduces the probability of this mistake, but makes the low accuracy problem even worse. Those problems make the votes of the same plane distribute sparsely in the 3D parameter space that makes the plane detection difficult and unstable.

Region growing based

Region growing based methods can be used for plane detection as well and many novel methods have been proposed in recent years. The common procedure is: first, some initial seed points are selected, then grown into regions based on the homogeneity of local features. The features can be chosen as color, normal direction, distance from the point to the plane, etc.. With the seed and the features, an efficient clustering method is applied to extracting all the potential planes.

In Weingarten et al.'s work [32], a point cloud is divided into several small cubes. For each cube, co-planar points are found by RANSAC. Then the least squares method is used to fit an optimal plane among these points. Subsequently, planes are extracted by merging small patches into big ones. However, the neighborhood information, that is, the relationship between neighboring cubes is ignored, which causes the result very coarse. Moreover, using RANSAC in each cube makes this method time consuming.

In Lin W's work [33], an unsupervised planar segmentation algorithm based on multidimensional (MD) particle swarm optimization (PSO) was proposed. MD-PSO algorithm is adopted to optimize an objective function proposed for planar segmentation. Meanwhile, a modification strategy of finding the global optimal position of the swarm in each dimension is added to the MD-PSO algorithm. Results showed that this method gained high accuracy of planar segmentation. However, this method is not appropriate for our research according to that it is only used for planar segmentation so that no information denotes the one belonging to the ground plane. The result is shown in Fig.1.2.

Two types of plane detection algorithms based on region growing are given in J. Xiao's work[34]: point-based region growing and grid-based region growing. In the point-based method, one point is selected from the point cloud, then regarded as a new seed if its local appearance is planar while satisfying that at least six or more neighbor points within a predetermined distance exist. Then the new seed together with its neighbors will be marked as a new growing region. This region grows under a certain rule and will be assigned to be a new plane when the number of points exceeds a threshold. Different to point-wise, grid-based method first separates the point cloud into several cubes, then the cube having the smallest MSE of all the unidentified cubes

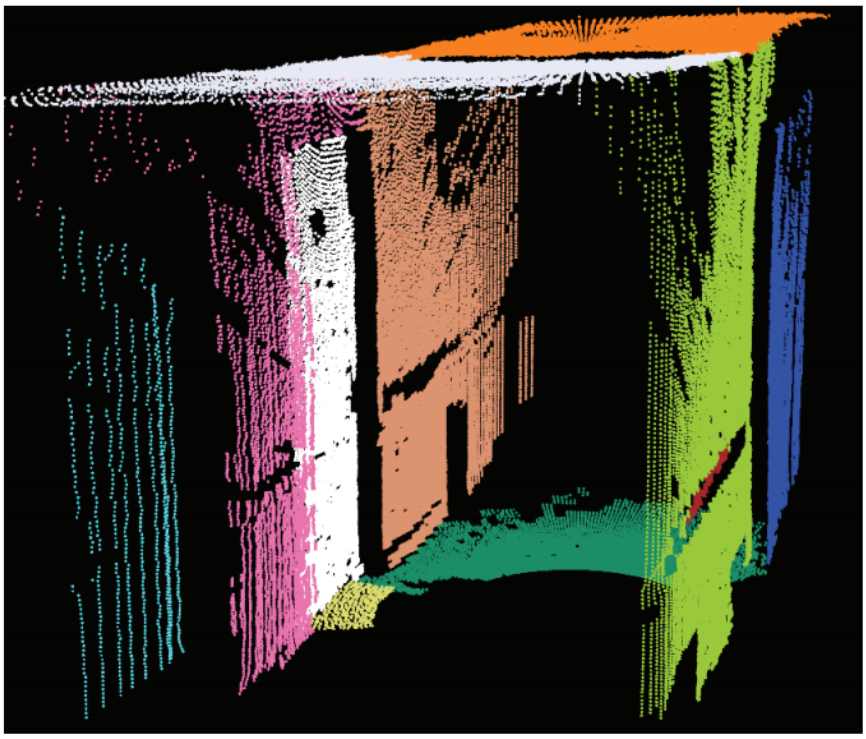


Figure 1.2: Planar segmentation result in research [33]

is selected as the seed. The region growing rule is similar to the one in point-based region growing. Grid-based method improves the computational speed dramatically.

Research in [35] proposed a method to extract planes from 3D range data captured by a range imaging sensor. Only vertical and horizontal planes from range images of indoor environment are considered to be extracted. Range images enhanced by the normal information are partitioned into several segments by using a method called Normal-Cut. Subsequently, each obtained segment is fitted by a least-square plane and the fitting error determines if the segment is a plane or not. For the resulting planar segments, based on the normal of its least-square plane, each segment is labeled as vertical or horizontal. Then neighbors with the same label will be merged. With this region growing processing, all the planes either vertical or horizontal can be detected.

Method in [36] uses polygons to reconstruct 3D models, where a new region growing algorithm of planar detection has been proposed. Starting with a pair of neighboring points, it repeatedly tries to extend the current set by examining all other points with the increasing distance from this point set. For a point closed enough to the current

set, it will be added to the current set only when both the average squared error and the distance from this point to the optimal plane are under the threshold. This process is repeated by choosing different seeds until all possible planes are detected.

Based on the work [37], Jann Poppinga made a modification to reduce the computational cost. In the region grow procedure, instead of examining only one point per iteration, an 8-neighbor structure is defined to determine the nearest group of points, which enhanced the efficiency dramatically. Moreover, calculations for the average squared error and distance from the point set to the optimal ground has been simplified. This method is fast but sensitive to noisy data.

Rabbani in [38] proposed a method of smooth area detection that can be used for plane detection. The normal of each point is estimated first, then point with the minimum residual is considered as the seed. For the last added point, k nearest neighboring points are used for region growing. However, in this method, angle between the normal of the point and the current normal of the plane is used as a constraint and only if it is under a predetermined threshold, the points will be added to the current set.

In Harati's work [39], with the pixel-neighborhood information, a measure called Bearing Angle is used for each point to evaluate the flatness of its local area. Based on this measure, a line-based region growing algorithm is proposed. However, since Bearing Angle is the incident angle between the laser beam and edges of the scanned polygon in the selected direction, it cannot be properly calculated in cluttered environments. Georgiev et al. in [40] proposed a method started by extracting 2D line segments from each 2D scan slice (each row or column in an organized point cloud), where connected line segments represent candidate sets of co-planar segments. Then, a region growing algorithm is utilized to find co-planar segments and their least squares fitting (infinite) plane.

In J. Xiao's work [41], a plane segmentation method called Subwindow Based Region Growing (SBRG) is proposed for plane segmentation from structured environment. The point cloud is decomposed into several subwindows, which are classified as planar and non-planar based on the method presented in this work. Subsequently, among the subwindows classified as planar, the one with the minimum mean square error is chosen as a new seed. This method has good performance in structured environments. However, the result is unsatisfactory in the case of unstructured environments. To deal with this problem, another algorithm is proposed called Hybrid Region Growing. In the

region growing procedure, if the neighboring subwindow is planar, the following steps are the same as those in SBRG. Otherwise, each point in the subwindow needs to be investigated separately then added into the region only if the distance to the optimal plane and the MSE are both under the threshold. Using subwindow as the seed makes growing step much faster than the pointwise method.

1.1.2 Problems of the present algorithms

After introducing and analysing the present algorithms, we have found several problems that almost all those algorithms are suffering from: First, the RANSAC based algorithm [11]-[14] are unable to detect multiple planes. Even though the extensions to RANSAC have been proposed for detecting multiple planes in [16, 17, 18, 19], they are more suitable to be used in planar segmentation, where no clear information indicates which planes belong to the ground.

Second, for the region growing based algorithms, since the performance is mainly dependent on the accuracy of seeds, the defect is apparent that, in a 3D scene, if a ground plane is divided by an obstacle into several parts, all those parts will be detected individually and considered as different planes. Besides that, the complexity of these algorithms will lead to numerous computational cost.

Third, in the case of ground plane detection, the camera pose is an important factor. Most existing studies did not take the camera pose into account. However, in many real applications, the camera pose is available although inaccurate. For examples, the camera mounted on a mobile robot often has a known pose and mobile phones have built-in gyroscope sensors. Applying the prior knowledge of the camera pose is possible to make ground plane detection fast and stable. Even though camera calibration was used in work [9] to calculate the depth of the ground plane, it didn't give any support to refine the result. Moreover, there is no multiple ground plane detection mentioned in this work.

1.2 Scene labeling

In computer vision, object classification or labeling is another meaningful and important task. The motivation is to predict the class of a specific object correctly from a 2D image or 3D scene. A typical application is to detect pedestrian from an indoor or outdoor

scene. The complexity of the background increased the difficulty of this task. However, labeling the scene to distinguish "background" and "foreground" objects, then using a effective classifier to detect the pedestrian only from the "non-background" objects could simplify the problem. Apparently, the performance of the detection depends on the accuracy of the labeling.

Generally, two components in a classification approach need to be defined: a set of feature vectors and a probabilistic graphical model. The extracted feature describes the characteristics of the object, which enables the model to predict the probability of each label. An effective model can compensate for insufficient features, and discriminative features can compensate for a too simplistic model.

In recent years, many scientific works focus on developing a robust and strong feature as well as the probabilistic graphical models. We will introduce those studies and discuss the existing problems respectively.

1.2.1 Literature review of scene labeling

Development of feature

Recently, various features used for scene labeling have been developed. In N.Dalal's work [42], a local feature called HOG (histogram-of-gradients) describing the appearance of person has been proposed. The method is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. It is inspired by the fact that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions. In the implementation, the image is divided into several small spatial regions, which are called as cells, then for each cell, a local 1-D histogram of gradient directions or edge orientations over the pixels is calculated. The cells are grouped into larger spatial blocks and contrast normalizing is implemented for them. The final descriptor is then the vector of all components of the normalized cell responses from all the blocks in the detection window. Finally, the classification step is implemented by using a linear SVM classifier. The proposed feature gives a good result on 2D image.

In Peter Gehler's work [43], several methods have been proposed to combine a set of diverse and complementary features instead of using a single feature type. This work focused on solving the problem that the instances belonging to the same class usually

have high intraclass variability. Developing an invariant feature may solve this problem, however it is clear that none of the feature descriptors will have the same discriminative power for all classes. Since the classifier used in this work is the SVM classifier, several methods of kernel combination have been developed. Meanwhile, it gives an approach of learning the kernel combination and the parameters during the training phase of the algorithm. Results showed that the method yielded good performance on 2D images.

Recent advances in 3D sensing technologies make it possible to exploit geometric and structural features those can enhance object recognition and many relative works have been proposed [44, 45, 46, 47, 48, 49, 50, 51].

A variety of advanced 3D histogram based feature descriptors have been concluded in Behley's work [52] including: (1) Histogram of Normal Orientations in [53], which is a normal histogram storing the angle between the reference axis and a neighboring point; (2) Spin Images in [54], which is calculated by spinning a grid around the reference axis, where the grid cells collect or "count" the neighboring points; (3) Distribution Histogram by [55], which tries to capture the shape around a point from a designed cube; (4) Signature of Histograms of Orientations (SHOT) in [56], generated by calculating a histogram of normal orientations between the neighboring point inside each divided sector and the query point. After summarizing those 3D histogram based feature descriptors, from SHOT, a feature descriptor called Spectral Histogram is also proposed, which calculates three signature values from the spectral values of the points inside each sector region. Then, the space around a point is subdivided into different slices and shells. For every radial shell in a slice, a different scale of the point distribution is obtained according to a constraint. Comparative experiments give the results that Distribution Histogram, SHOT, and Spectral Histogram are the descriptors that resulted in the best performance.

In addition to that, many features have been proposed and widely used such as: spherical harmonic invariants [57], curvature maps [58],and conformal factors [59]. However, it has been proved that those features have a limitation to some application due to the properties of themselves. In recent years, Radu Bogdan Rusu in [60] has proposed a 16-dimensional feature vector called PFH (Point Feature Histogram), which describes the local geometry around each point p in a point cloud. PFH extracts four features [α, φ, θ, d], where α is the angle to the second axis, φ is the angle to the first axis, θ is a rotation on the UW plane and d is a distance between two points, which is

used for weighting parameter. A Point Feature Histogram representation is based on the relationships between the points in the k-neighborhood and their estimated surface normals. It attempts to capture the sampled surface variations by taking account all the interactions between the estimated surface normals.

In [61], a new type of local feature, called Fast Point Feature Histograms (FPFH), which retains most of the discriminative power of the PFH but has an improved calculation speed, have been developed. The FPFH feature vector is a simpler but faster version of PFH by catching previously computed values in feature histogram computation. The number of bins is set to be 11 for each α, φ and θ. Therefore the FPFH descriptor can be represented with 33 dimensional vectors. This feature has been shown to be invariant to position, orientation, and point cloud density, and it copes well with noisy datasets. However, for different categories of objects which have similar shape, this feature becomes weak to distinguish. In addition to that, FPFH is invariant to view-point.

Many methods focusing on developing global features such as Extended Gaussian Images (EGI) [62], eigen shapes [63], or shape distributions [64] have been proposed. EGI in [62] describes the feature of an object from the unit normal sphere, and it has the problems of handling arbitrarily curved objects. In [63], for each model, images in the training set are spanned into a large training matrix. Then using principle component analysis (PCA) to find k eigenvectors corresponding to the largest eigen values. These k eigenvectors form a special subspace for each model. The recognition step is done by projecting the test image into the trained subspace then using Nearest-Neighbor to find the best matching. Eigen shapes show promising results but they have limits on their discrimination ability since important higher order variances are discarded. The latter feature samples statistics of the entire object and represents them as distributions of shape properties, however they do not take into account how the features are distributed over the surface of the object.

In Radu Bogdan Rusu's work [65], a new feature called Viewpoint Feature Histogram (VFH) has been proposed. Based on FPFH, this feature encodes important statistics between the viewpoint and the surface normals on the object, which makes VFH a global feature corresponding to the view point. VFH is constituted by an additional component to FPFH, which is calculated by collecting a histogram of the angles that the viewpoint direction makes with each normal. Then, the FPFH component

measures the relative pan, tilt and yaw angles between the viewpoint direction at the central point and each of the normal on the surface.

Another useful feature named Pyramid Context Feature is developed by work [51]. This feature is designed for per-voxel linear classification. For each voxel V, a feature vector is calculated by vectorising the linearly-interpolated voxel, then a K-level Gaussian pyramid of V is computed. Each k-th pyramid context feature is upsampled with the corresponding scale and summed up together to form the final feature. Not only is this feature a powerful descriptive representation, but also is designed such that exact per-voxel linear classification can be made extremely efficient.

Arbeiter's work [46] chose three promising descriptors, namely Radius-Based Surface Descriptor (RSD), Principal Curvatures (PC) and FPFH, and presented an approach for each of them to show how they can be used to classify primitive local surfaces such as cylinders, edges or corners in point clouds. However, those features are neither sensitive to camera transformation, nor to the shape similarity.

Development of graphical model

Graphical model is a probabilistic model which expresses the conditional dependence between random variables and has been widely used for labeling. Recently, more and more significant works focus on developing graphical model to gain a better performance. In H.S.Koppulas's work [66], the author proposed a graphical model that captures various features and contextual relations, including the local visual appearance and shape cues, object co-occurrence relationships and geometric relationships. Both associative and non-associative coupling of labels have been modeled. The model is trained by using a maximum-margin approach that globally minimizes an upper bound on the training loss.

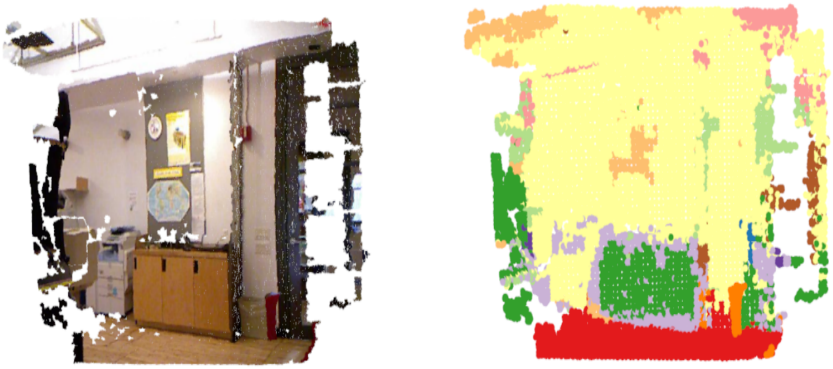
In M.Naja's work [67], the author introduced a non-associative higher-order Conditional Random Field (CRF) to address the problem of 3d point classification, in which four higher-order patterns (simple co-occurrence, geometric co-occurrence, within clique adjacency and height signature) were designed and formed together as a new pattern-based potential term. This model is invariant to the size and number of the segments after dividing the scene, and the labeling problem is solved by Maximum A Posteriori (MAP) estimation.

In Mayank B.'s work [68], an outdoor stereo image classification approach was proposed by defining a spatial 3D feature called Vertical Support Histogram (VSH) from dense stereo range maps to locally characterize 3D structure. Meanwhile, the context model describing the specific relationship between pair-wise labels is designed for using Markov Random Field. However, the feature vector in this work is only developed from the vertical distribution so that it is unable to exploit the spatial information effectively. Meanwhile, the context model has some limitations since it is manually defined. The result of their work is shown in Fig.1.3.

Figure 1.3: Scene labeling result in research[68]

In Daniel Wolf's work [69], an efficient semantic segmentation framework of indoor scenes operating on 3D point clouds has been proposed. This work used the results of a Random Forest Classifier to initialize the unary potentials of a densely interconnected Conditional Random Field, and learned the parameters for the pairwise potentials from the training set. These potentials capture and model reasonable spatial relations between class labels. Features extracted from each voxel consist of eigenvalues, angle with ground plane, height and color in CIELAB space. This approach has achieved a significant result while holding a fast computation speed. Result of this work is shown in Fig.1.4.

Silberman et al. presented the first large indoor RGB-D dataset [70], containing thousands of frames, from which over 2,000 have been densely annotated. They train



(a) input points cloud (b) labeling result

Figure 1.4: Input points cloud and the labeling result in research[69]

a neural network as a classifier and then apply a CRF model to label the indoor scene. However, class label relations were not completely considered in this research. Ren et al. [71] proposed another indoor labeling algorithm on the same dataset. Comparing with the traditional pointwise approach, they divide the point cloud into several voxels, then use kernel descriptors to describe the patches. The labeling is solved by using Markov Random Field (MRF) with a segmentation tree. However, the complexity of their approach leads to large computational cost.

Valentin et al. [72] use an over-segmentation task for a mesh representation of the scene. Each mesh is classified by a JointBoost classifier, then use Conditional Random Field to obtain the labeling result. But label relations were taken into account either. Hermans et al. [73] proposed an indoor scene labeling framework based on Random Forest classifier and a dense CRF model. Random Forest achieved a better classification result by comparing with other classifiers. However, the feature they used is extracted from 2D data. Moreover, the potential is modeled as a simple Potts model which is not accurate enough to describe the label relations.

The label relations have been incompletely considered in those researches above. Even though the features they used are powerful, the defect brought by the independency is inevitable. To solve these problems, several methods have already been proposed for the purpose of learning and modeling the contextual relation between class labels.

Anand et al. [74] proposed a huge graphical model for solving the labeling task,

which can capture the spatial relations of class labels depending on different features. However, the complexity of this approach makes the computation speed very slow. Kahler et al. [75] combine the binary tree with CRF for scene labeling. In this work, unary and pairwise terms of CRF are learned through two methods, the Decision Tree Fields and the Regression Tree Field, respectively. The DTF inference is done by using a Gibbs sampler since it is a discrete optimization. For RTF inference, an efficient, exact inference method proposed in [76] is applied.

A framework with a voxel-CRF model is proposed in Kim et al's work [77]. In this work individual voxel observation is modeled as the unary term. Pairwise term is modeled to capture the label relations between the neighboring voxels. In additional to that, higher-order potential that encodes the relationship among more than two voxels are also proposed in order to capture the label relations within a group of voxels. This novel design enables itself to extract numerous information of label relations. However, computational cost also increases with the complexity of the model.

1.2.2 Problems of the present algorithms

After comparing the performance of the present algorithms, we give a conclusion about the problems these algorithms are suffering from:

1. **Feature**: In recent years, many state-of-the-art studies relating to 2D scene labeling, segmentation and classification have been reported in literature and many discriminative features have been developed. However, features used in those researches are typically developed by capturing the 2D appearances of the objects. Developing a feature capturing the 2D appearance of object needs multiple scales to be considered. Since no scale information is available in 2D image, this will lead to more failure results. Meanwhile, a lot of valuable information such as the geometry distribution and the layout of object, cannot be extracted from 2D image.

Development of 3D sensor makes it possible to exploit geometric and structural information such as 3D position, appearance and the orientation of surface. However, geometric feature, which is used for describing the structural characteristic of 3D objects and considered as a very useful feature, has not been exploited effectively. For instance, some studies use height distribution as the feature vector, which is attempted

to distinguish the objects from the difference of their height. However, in the patch-wise or voxel-wise algorithms, using this feature will cause failure since some voxels belonging to different categories always holding the similar height distribution.

2. **Label relation**: For many indoor environments, scenes usually exhibit very distinctive structures and repetitive spatial relations between different categories of objects. For instance, person is always standing on the ground and under the roof, and other objects should never be under the ground. When the features are not sufficiently discriminative to predict labels correctly, exploiting spatial relation of labels can significantly improve the performance. In some studies where label relations are not taken into account, even though a robust and strong feature is used, the performance always suffers from blocking effect caused by independency. Others use graphic model to solve this problem, where Potts model is a common option for modeling the pairwise potential according to the consideration that neighboring objects with the similar features (color, normal and etc.) are more likely to have the same label, while this consideration is not applicable for all scenes. Overall, relationship between different labels have not been learned sufficiently.

3. **Rotation invariance**: Even though many strong features have been developed, rotation invariance is rarely considered in those research. It is very common that the camera coordinate of a scene captured by a mobile robot is changing all the time, either horizontally or vertically. The commonly used features such as position and normal are very sensitive to rotation. In recent years, many useful features describing the appearance which are insensitive to camera rotation have been developed. However, in our research, these features are not strong enough to distinguish the objecting with a similar planar appearance (Roof and Ground). It is necessary to develop a new feature not only discriminative, but also robust to camera rotation.

1.3 Purpose of this thesis

The purpose of this thesis consists of two parts:

The first part is to set up a framework for detecting either single or multiple ground planes in 3D point cloud. The ground planes are allowed to be parallel or non-parallel. Also, in this framework, the camera tilt angle is estimated and used as the feedback to refine the detection result iteratively.

The second part is to set up a Bayesian framework for 3D indoor scene labeling. For practical needs, labels in this thesis consists of Roof, Wall, Person Candidate and Ground (R, W, C, G). Our attemp is to develop a set of features that can exploit rich partial information from a full-scene cloud point and is robust against camera rotation. In addition to that, we aim to model the label relations correctly. Finally, we attempt to detect the person from the objects labeled as Person Candidate.

1.4 Overview of this thesis

Chapter 2 gives a framework of detecting single or multiple ground planes with the estimation of the camera tilt angle. The contributions of this algorithm includes:

1. In this algorithm, a method called θ-projection is proposed. We use this method to select the points belonging to the ground plane with a parameter, which is represented as the camera tilt angle. Another method for estimating the camera tilt angle from the selected points is proposed. We use both of the methods iteratively to refine the results of the detected ground plane and camera angle.

2. We give a modification of the previous algorithm in 1 to improve the accuracy and running speed.

3. Based on the algorithm in 2, we make a modification on it so as to detect both parallel and non-parallel ground planes.

4. We compare our algorithm with other state of the art works to evaluate the performance.

Chapter 3 gives a Bayesian framework based on Markov Random Field that labels the 3D stationary indoor scene. The contributions of this algorithm includes:

1. We propose a method to learn a novel spatial feature vector that derives from the combination of three directional distributions by using eigenvector decomposition and sub-space combination.

2. We model both associative and non-associative coupling of labels that accommodates the 3D point cloud, then develop a corresponding smooth cost function that is used for Markov Random Field.

3. The labeling is obtained by solving the MAP estimation. In this thesis, we use a linear combination of cost energy so that MAP estimation is equivalent to minimizing the cost energy.

4. The final result suffers from the uncertainty brought by the size of the divided cube, which will be discussed in the experimental part in this chapter. This problem will be solved by using the approach proposed in chapter 2.

Chapter 4 gives another Bayesian framework that labels the 3D indoor scene with camera rotation. The contribution of this algorithm includes:

1. To solve the problems of camera rotation, we develop a set of feature vectors which is discriminative and robust to camera rotating based on the distribution of the normal vector.

2. To overcome the defects brought by MRF, we choose to use Conditional Random Fields to solve the labeling problem. Meanwhile, we propose a method to learn the label relations between two different categories of objects rather than manually defining it, then use the learned model to calculate the pairwise potential of the CRF.

3. We model a new pairwise potential of the CRF to improve the labeling performance.

4. We develop another feature used for detection persons from the results obtained by our labeling algorithm.

In this thesis, each chapter provides the experimental results and conclusions. In Chapter 5, we give a total conclusion and discuss the future work.

2

Ground Plane Detection

In this chapter, a single/multiple ground planes detection algorithm where camera tilt angle estimation is utilized as an assistant for improving the accuracy is proposed to solve the problems discussed above. The main motivation of this research is to use the acquisition of camera tilt angle to make the detection of ground planes fast and stable. The idea is, in an indoor scene captured by a RGB-D camera, height of the points on the ground plane will be nearly the same. If we project all the points in the whole scene onto a line parallel to the estimated normal vector of the ground plane (corresponding to the camera pose), the height distribution will become concentrate and show a high peak denoting the location of the ground plane. Thus we can use a voting mechanism similar to the Hough Transform to detect ground planes. In our case, the votes will distribute more densely than that in the 3D Hough Transform space. This makes our method much simpler and need lower computations. Since the available or assumed camera tilt angle maybe inaccurate, after the ground plane is detected, we can use the result to refine the camera pose, then detect the ground plane iteratively to improve the accuracy.

In order to achieve a robust performance of detection, the following key ideas are used in this thesis:

1. Applying a height distribution based method to detect the location of the ground planes with an estimated camera tilt angle.

2. Defining an appropriate region to select the points belonging the ground planes.

3. Using a method to refine the ground planes and the angle iteratively.

The content of this chapter is organized as follows:

1. In subsection 2.1.1, we give an overview of our algorithm of single ground plane detection.

2. A method is proposed in subsection 2.1.2 for the purpose of finding the location of ground plane under an assumed camera tilt angle. Then based on the location, a voting rule for the ground points is proposed.

3. Based on the method in subsection 2.1.2, we proposed an approach to find a roughly estimated camera tilt angle in subsection 2.1.3.

4. In subsection 2.1.4, we show how to refine the ground plane and the camera tilt angle iteratively.

5. In subsection 2.1.5, we make some modifications to the method proposed in 2.1.2 to improve the accuracy.

6. In section 2.2, we use our algorithm to detect multiple ground planes, including parallel and non-parallel ones.

7. In section 2.3, we give the details of our experimental results. We also make comparisons with other algorithms.

8. In section 2.4, we give a short conclusion and discussion.

2.1 Single ground plane detection

2.1.1 Overview of our algorithm

In this work, we assume that the height of the ground points and other points are independent. Meanwhile, only camera tilt angle is taken into account. Figure 2.1 shows the flow chart of our algorithm. The whole algorithm can be divided into two procedures: Rough Detection (RD) and Precise Detection (PD). For an input point cloud, before any procedures, a preprocessing is applied to project all the points onto a height corridor proposed in [78]. This height corridor consists of three parts: high, middle and low regions. Points within the low region are considered as belonging to the ground plane. We use this corridor to discard the points belonging to other objects

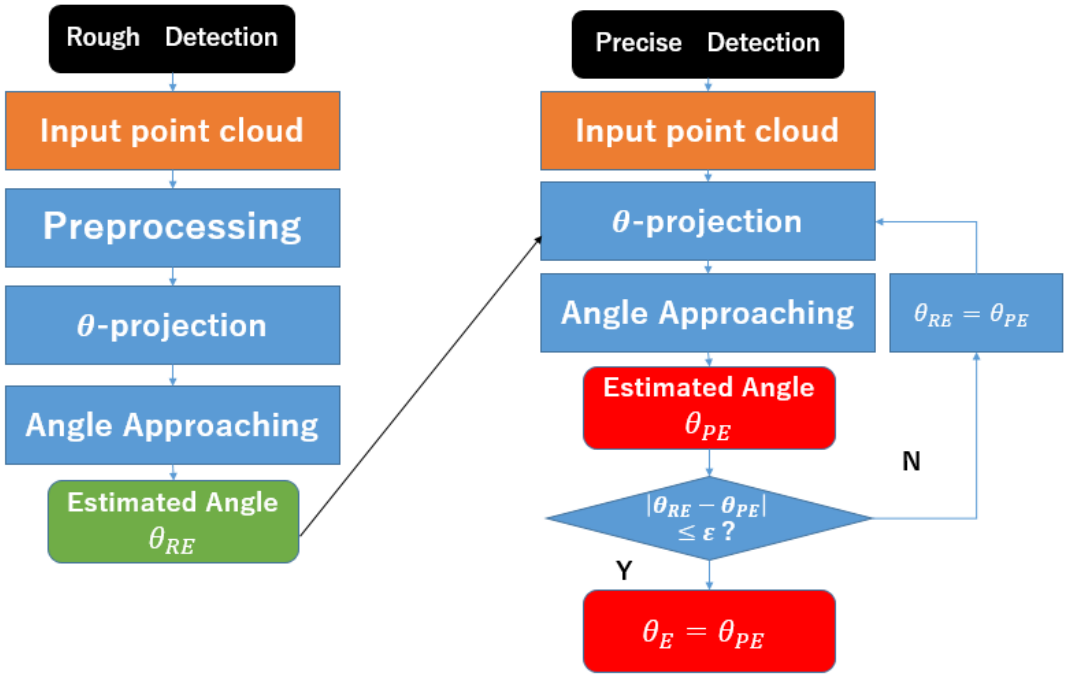


Figure 2.1: The figure shows the framework of our algorithm, where Rough Detection gives a roughly estimated angle θ_{RE}, and output of Precise Detection is the estimated camera tilt angle θ_{PE}. The ground plane is detected based on θ_{PE}.

so that the number of ground points will be the largest. However, if the camera tilt angle is a big negative value, some part of the ground plane will span out of this region and then mistakenly be discarded. For this reason, in our work, low region of this height corridor is enlarged to ensure that all the ground points can be contained in it. Remember that this preprocessing will only be carried out for single ground plane detection.

After the preprocessing, in Rough Detection stage, a rough range centered at 0 degree is set. Then for each angle in this range, a method called θ-projection, (described in 2.1.2) is used to select a set of points belonging to the ground plane hypothetically. Subsequently, a method called Angle Approaching finds the most appropriate angle through those sets. However, this angle is not accurate enough to be considered as the final result.

In Precise Detection procedure, we define another range centered at the roughly estimated angle from RD stage. Note that this range is much smaller than that defined in RD stage. The most accurate angle in this range is chosen through Angle Approach-

ing as well and refined by comparing with the former one iteratively. Finally, ground plane is detected under this angle.

Details of each procedure will be discussed in the following parts.

2.1.2 θ-projection

As the core of the whole algorithm, a method named θ-projection is proposed in this work. This method aims to create a set of points considered to be the ground plane under a given camera tilt angle. Our scene is represented as a point cloud captured by a RGB-D camera. In the real world, a signature difference between the ground plane and other objects is that the height of ground points is nearly the same. This feature can be clearly observed from the height distribution. The shape of height distribution has two characteristics: first, a sharp peak exists in the height distribution, which indicates the location of the ground plane. Second, the variance of the distribution is very small, indicating that the ground points are collectively distributed. Generally, according to the height distribution, if the location of the ground plane is found, which is represented as the height corresponding to the peak, then detecting the ground plane will be much easier since the ground points are concentrated at this height within a narrow range. For this reason, equalization of the height distribution is the principal work in this algorithm.

Kernel Density Estimator

Kernel Density Estimation (KDE) is a non-parametric method to estimate the probability density function of a random variable. The definition of KDE is shown below:

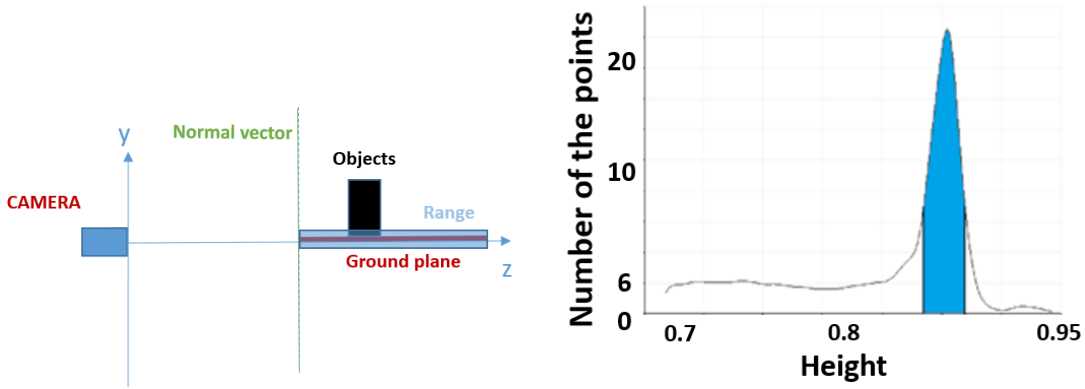
For a set of independent random variables $\mathbf{x} = \{x_1, x_2, ..., x_n\}$, each sample x_i is drawn from some distribution. Its kernel density estimator is used to estimate the unknown density p of the distribution, which is given by:

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \tag{2.1}$$

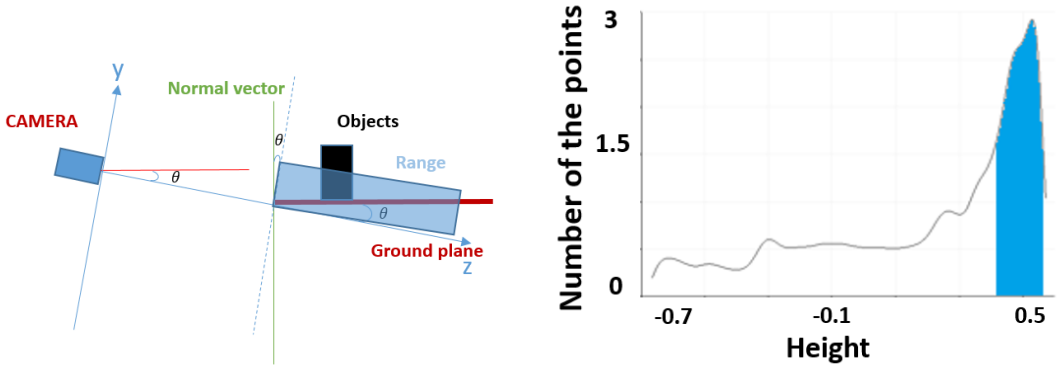
where the kernel $K(\cdot)$ is a non-negative function centered at x_i and integrates to one. h is called the bandwidth. If we use Gaussian kernel, KDE of the height distribution can be expressed as:

$$p(y) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}w} \exp\left(-\frac{(y - y_i)^2}{2w^2}\right), \tag{2.2}$$

where w is the bin width of the height histogram. y_i is the height of each point.



(a) height distribution when Camera tilt angle is zero and the corresponding range



(b) height distribution when Camera tilt angle is zero and the corresponding range

Figure 2.2: This figure shows the distributions when the camera tilt angle is zero and nonzero. In (a), most of the points in the range (illustrated as blue color) belong to the ground plane. In (b), some parts of the object on the ground are wrongly falling into the range.

After we equalize the height distribution, the ground points can be found from an appropriate range centered at the peak as shown in Fig.2.2 (a). However, this theory only works in the case that the camera tilt angle is zero. In practical, this way is not applicable since camera tilt angle is always nonzero. Consider the situation where the camera angle is nonzero as shown in Fig.2.2 (b), the optic axis (Z axis) of the camera is not perpendicular to the normal of the ground, which turns out that the height of each point in the camera coordinate system is no longer equal. Therefore, if we use the corresponding height distribution straightly, the range centered at the peak is

inclined to contain not only the points belonging to the ground plane, but also those belonging to other objects, which will affect the result. However, if the camera tilt angle is predictable, this problem can be solved by camera calibration.

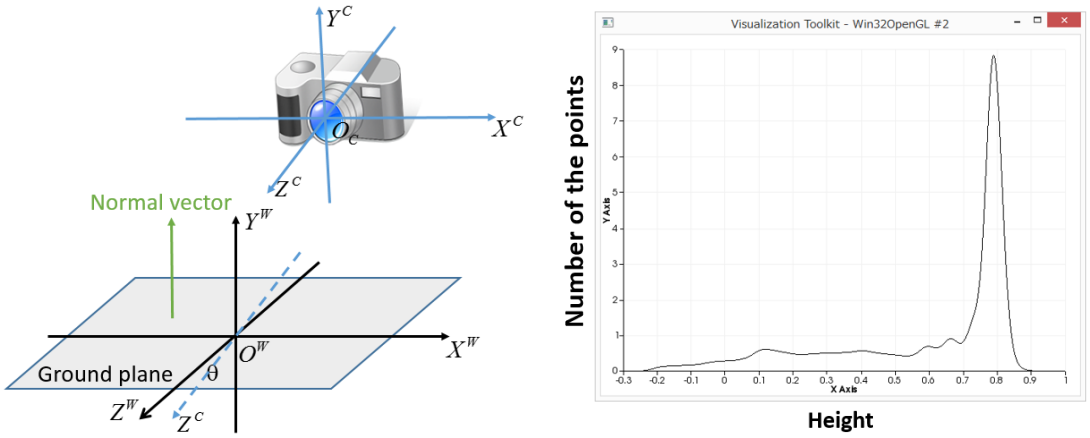


Figure 2.3: Height distribution after projecting the ground plane onto the normal vector when the camera tilt θ is known

Distribution after projection

As shown in the left part of Fig.2.3, the ground plane in the world coordinate system (X^W, Y^W, Z^W) is perpendicular to axis Y^W. Without considering the roll angle, if the camera tilt angle θ is non-zero, in the camera coordinate system (X, Y, Z), the angle between the normal of the ground plane and the horizontal axis will be θ as well. We aim to find the height distribution where the ground points are extremely concentrated. If the camera tilt angle is known, this can be achieved by projecting the 3D points onto the assumed normal vector (corresponding to θ). The corresponding height distribution is shown in the right part of Fig.2.3. For a given point cloud $P = p_i$, the height y_{θ_i} of each point p_i after being projected under an angle θ is calculated by:

$$y_{\theta_i} = y_i \cos\theta - z_i \sin\theta, \tag{2.3}$$

where $i = 1, 2, \ldots, n$, y_i and z_i is the height and depth of p_i in the camera coordinate system, and n is the number of points in the point cloud. The range $[H_{min}, H_{max}]$ of the heights $\{y_{\theta_i}\}$ is divided into N bins equally, so the bin width w is calculated by:

$$w = \frac{H_{min} - H_{max}}{N}, \tag{2.4}$$

where H_{min} and H_{max} are the minimum and the maximum heights respectively. Then the KDE of the height distribution after projection under the angle θ is given by:

$$p(y_\theta) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\sqrt{2\pi}w}\exp\left(-\frac{(y_\theta - y_{\theta_i})^2}{2w^2}\right), \tag{2.5}$$

where we have chosen Gaussian Kernel for $K(\cdot)$ to solve the artificial discontinuities problem according to the conclusion in [79]. The next step is to define an appropriate range from this distribution for selecting the ground points.

Definition of the range

After the projection, we consider the points within a range centered at the peak of the distribution as belonging to the ground plane. We use

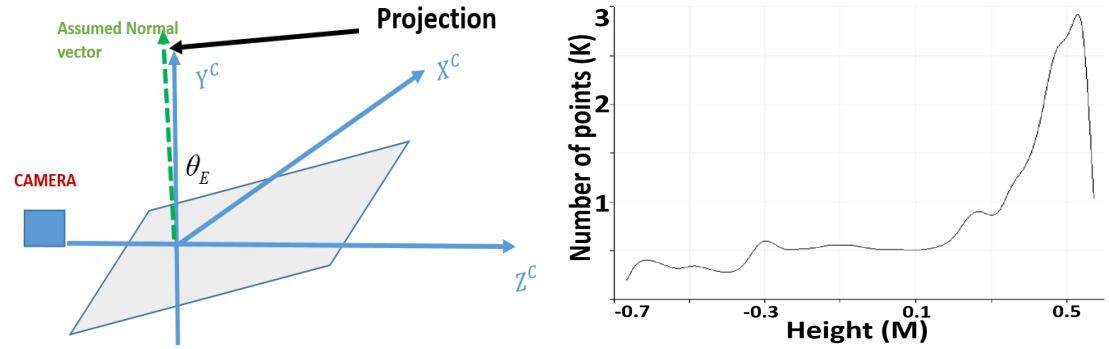
$$\Delta = [y_{max} - \tilde{\sigma}, y_{max} + \tilde{\sigma}] \tag{2.6}$$

to express that range where $\tilde{\sigma}$ is a parameter determining the size of Δ. y_{max} is the height maximizing $p(y_\theta)$, that is

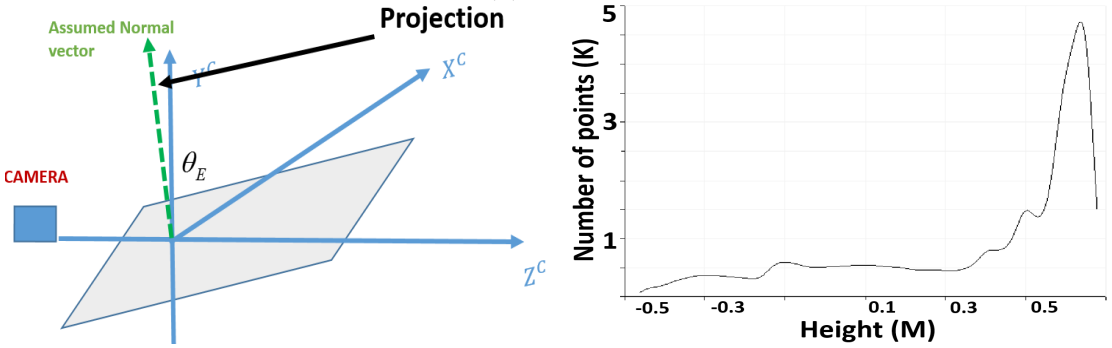
$$y_{max} = \underset{y_\theta \in [H_{min}, H_{max}]}{\arg\max}\ p(y_\theta) \tag{2.7}$$

In order to make Δ contain more ground points and less noise points (those belonging to other objects), it is important to determine a reasonable value for $\tilde{\sigma}$. According to the experimental results in this work, we find that a good result can be obtained when $\tilde{\sigma}$ equals to the standard deviation of the height distribution. A higher value of $\tilde{\sigma}$ will make more noise points fall into Δ, and a smaller value will result in a decrease in the number of ground points within this range. Either of them will affect the result.

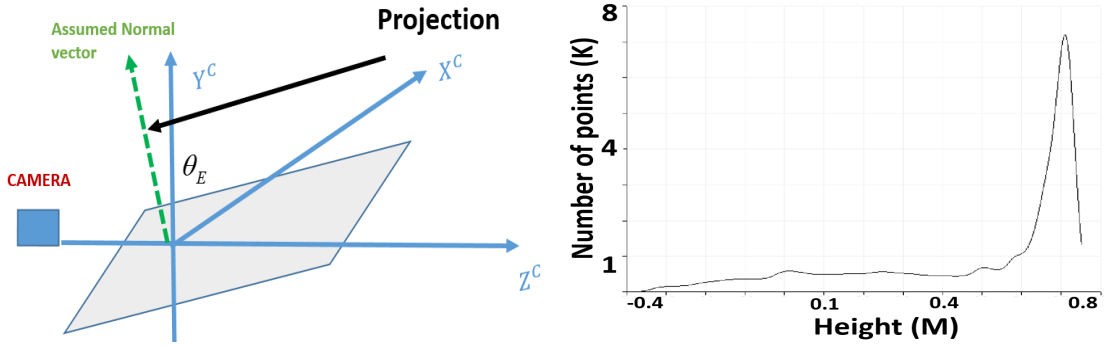
The points falling within Δ are considered as the ground points, and we call this algorithm as θ -projection. Apparently, the unknown camera tilt angle is the key factor in this method. As shown in Fig.2.4, if the assumed tilt angle contains some errors, points on the ground plane will distribute in a wider range. With the assumed angle closing to the true value, the shape of the distribution is getting sharper. According to this feature, we use θ-projection as a core method in both RD and PD for estimating the angle and finding the best set of ground plane points.



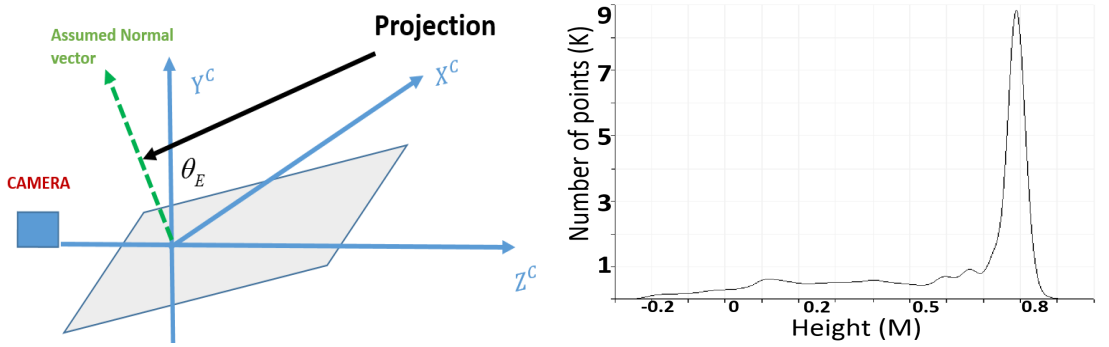
(a) $\theta_E = 0°$



(b) $\theta_E = 5°$



(c) $\theta_E = 10°$



(d) $\theta_E = 15°$

Figure 2.4: Height distributions (right part) after being projected by different angles. θ_E is the estimated angle. $\theta = 15°$ is the ground truth. The left part shows the ground plane in the camera coodinates (represented as Y and Z axis).

2.1.3 Rough detection (RD)

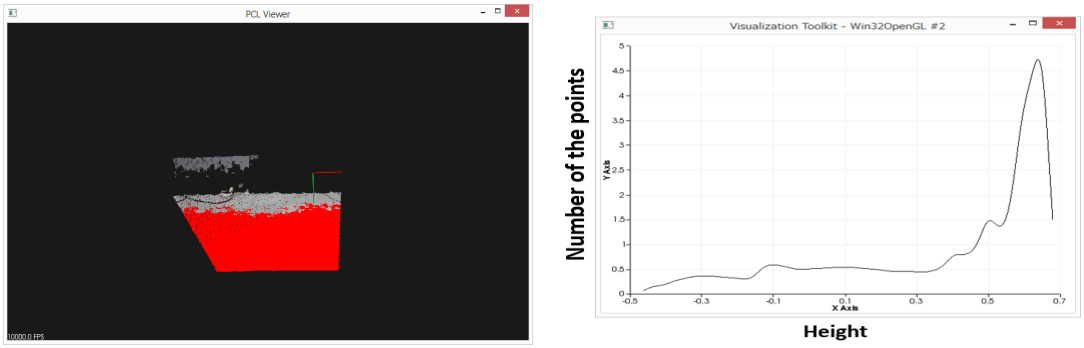
In this procedure, for the original point cloud, after the preprocessing, numerous points not belonging to the ground plane have been discarded. Since the camera tilt angle can be roughly predicted, we use this prior information to set an appropriate angle range in which the true value may exist. For example, if the camera is a Kinect camera and set horizontally (on a table or floor), this range can be set as $[-37°, +37°]$, which is the whole tilt range of Kinect camera. However, the maximum bound of this range should be limited within $+45°$. This limitation is reasonable since in common, the ground plane cannot be captured if the camera tilt angle is over this angle. With each integral angle θ_i in this range, we perform θ-projection and then create a set of ground hypotheses $\{G_i\}$. As we discussed above, the camera tilt angle is estimated according to the shape of the height distribution after projection. Theoretically, this can be achieved by observing the variance of the ground points since it is getting smaller with the estimated angle approaching to the real value. However, the uncertainty of the ground points makes it difficult to implement. Meanwhile, the probability density calculated by the KDE describes the height distribution of all the objects in the scene. Particularly, in a complex scene, the variance of the height changes with no rules. This phenomenon will be explained in subsection 2.1.5.

Instead of using the variance, we can estimate the angle from the perspective of probability. The output of θ-projection G_i under an estimated angle θ_i includes parts of the ground points and the noise points (points belonging to other objects). When angle is estimated correctly, more and more ground points will be selected while the number of noise points is decreasing. Ground points will be completely selected when the angle equals to the true value since they are the most convergent, which makes the number of G_i reach the maximum. This is illustrated in Fig.2.5, where red color shows the selected points.

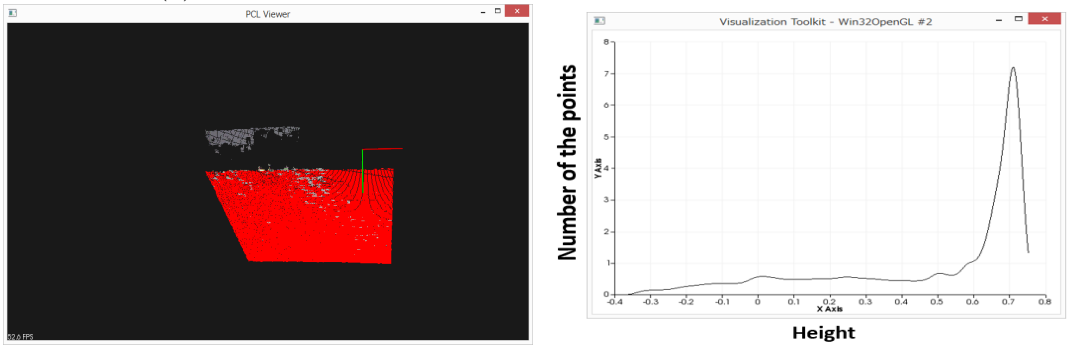
Since each points set G_i is created from the same points cloud, the camera tilt angle can be estimated by the proportion of the number of G_i . This proportion is given by:

$$F_\theta = \frac{N(G_i)}{N(G_o)} \tag{2.8}$$

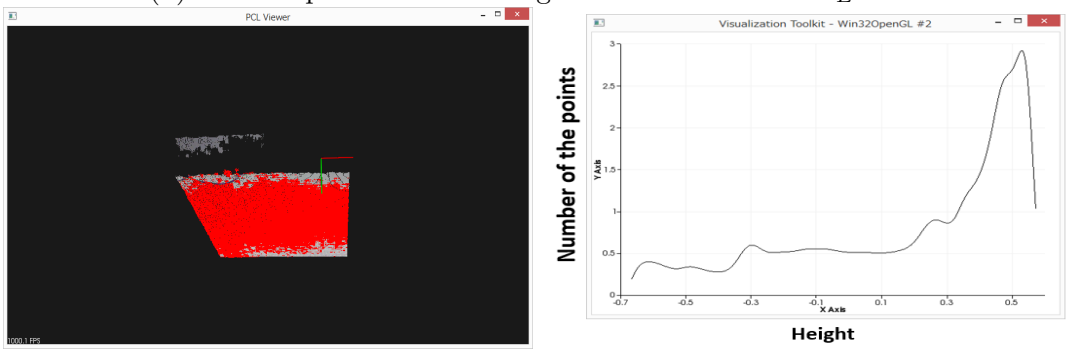
$N(\cdot)$ is used for calculating the number of points, and G_o is the input of the Rough Detection procedure. The angle that makes the proportion F_θ reach the maximum will



(a) selected points and the height distribution when $\theta_E = 8°$



(b) selected points and the height distribution when $\theta_E = 10°$



(c) selected points and the height distribution when $\theta_E = 12°$

Figure 2.5: The left part shows the seleted points which are illustrated by red color. All the selection procedures are implemented on the same point cloud. The right part shows the corresponding height distributions. The true value of the camera tilt is 10°.

be viewed as the output of this procedure, that is

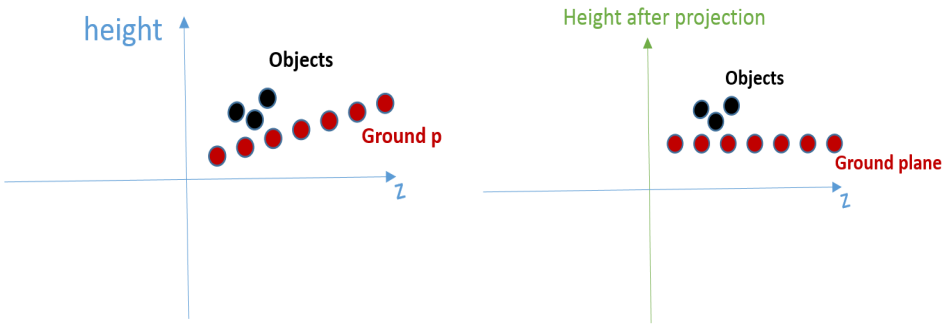
$$\theta_{RE} = \arg\max_{\theta_i} F_{\theta_i} \tag{2.9}$$

In practical, θ_{RE} is still not enough to be confirmed as the true value. Since G_i is constituted by the points falling into the range defined by eq.(2.6), there is inevitably some noise existed so that using this angle to detect the ground plane will cause a bad result. More specifically, considering two points sets G_a and G_b, which are obtained by using θ-projection with angle a and b. b is the true value of the camera tilt angle, $a \neq b$ but a is very close to b. G_b only contains the ground points, while G_a contains most part of the ground points and some noise points. In some cases, if there are too many noise in G_a so that the number of G_a exceeds that of G_b, angle a will be the output rather than b. According to this reason, result brought by θ_{RE} is not accurate enough. However, the information we obtained through this procedure is that the true value is very close to θ_{RE} . By implementing the procedure PD introduced in the next subsection with this angle, we can find both the ground plane and the most accurate value of the camera tilt angle.

2.1.4 Precise detection (PD)

In this procedure, we focus on estimating the tilt angle more precisely with the result of RD procedure. First, we need to get a refined set of points G_{RE} where the number of the ground points is much larger than that of the noise. G_{RE} is obtained by running θ-projection on the G_o with angle θ_{RE}. After that, with a small bias θ_ϵ, we define a range centered on θ_{RE} as $[\theta_{RE} - \theta_\epsilon, \theta_{RE} + \theta_\epsilon]$, similar to RD, each angle θ_{P_i} in this range is used for running θ-projection on G_{RE} to create a new set of hypotheses $\{G_{P_i}\}$. Alternatively, in this time, we do a little modification on θ-projection to make $\{G_{P_i}\}$ contain as less noise as possible so that the angle can be correctly estimated. Since each angle in the defined range is very close to the true value, and most points in G_{RE} are belonging to the ground plane, after projecting all the points of G_{RE} onto the normal of the ground plane, the height $y_{\theta_{P_i}}$ of each point calculated by eq.(2.3) will concentrate to a certain value, as Fig.2.6 illustrated.

For this reason, instead of using Δ in defined by eq.(2.6), we use K-Means to filter the noise and make the obained set of hypotheses $\{G_{P_i}\}$ contain pure ground points. Details of this procedure are given as follows: for each angle θ_{P_i},



(a) before projection (b) after projection

Figure 2.6: This figure shows the ground points and the noise points before and after being projected onto the normal vector. The height of each point is almost the same, based on which we can use K-Means to filter out the noise points

Step (1): project each point in G_{RE} onto the normal vector corresponding to the estimated angle θ_{P_i}, then calculate the height of each point $y_{\theta_{P_i}}$ by using eq.2.3, after that, calculate the mean height $\bar{y_{\theta_{P_i}}}$.

Step (2): With a threshold, calculate the distance of each point to $\bar{y_{\theta_{P_i}}}$. Point corresponding to the distance that is over the threshold will be discarded.

Step (3): For the remaining points re-calculate $\bar{y_{\theta_{P_i}}}$ and repeat Step (2). After N times, the remaining points constitute the hypothesis set $\{G_{P_i}\}$.

In this time, noise in each set G_{P_i} is very little. It is reasonable to utilize eq.(2.8) and eq.(2.9) to find the optimal angle θ_{PE} in that range. After that, θ_{PE} is compared with θ_{RE} and considered as the output θ_P if the constraint condition $\|\theta_{PE} - \theta_{RE}\| \leq \epsilon$ is satisfied, otherwise we make $\theta_{RE} = \theta_{PE}$ and repeat the same steps in PD. After θ_P is decided, points detected under this angle will be considered as the ground plane with the highest accuracy.

2.1.5 Modifications and improvements

We have proposed an algorithm used for detecting single ground plane, where camera tilt angle is an important factor to refine the results. In this algorithm, θ-projection is used in both Rough Detection and Precise Detection procedures. In θ-projection, after we calculate the probability density of the height distribution by eq.(2.3) and eq.(2.5), the deviation of this distribution is used to define a range Δ by eq.(2.6) for the purpose

of selecting the points belonging to the ground plane. Unfortunately, it is no doubt that some noise points(that not belong to the ground plane) will be selected, the number of which has an impact on the performance of our algorithm. An reasonable size of this range is expected to force more noise points discarded so as to gain a better result. However, since the distribution varies with different angle, the range defined by the deviation is unstable. This could be explained as follows

1). In a 3D scene, for all the 3D points $P = \{P_G, P_O\} = \{p_i\}$, where P_G and P_O represent the ground points and the points of other objects respectively, and $p_i = (x_i, y_i, z_i)$, $Y = \{y_i\}$ and $Z = \{z_i\}$ are their heights and depths before projection. $Y^{'} = \{y_{\theta_i}\}$ is the height calculated by eq.(2).

2). the variance of $Y^{'}$is calculated by

$$Var(Y^{'}) = \cos^2\theta Var(Y) + \sin^2\theta Var(Z) - \sin(2\theta)Cov(Y,Z) \tag{2.10}$$

3). If P only contains ground points, that is $P = \{P_G\}$, and each p_i satisfies a plane function $y_i = \tan\theta_T z_i + b$ strictly (since only the camera tilt angle θ is considered, we use a function of line to express the plane), where θ_T is the ground truth of the camera tilt angle, so Y and Z are linearly dependent, which makes:

$$Var(Y) = \tan^2\theta_T Var(Z), Cov(Y,Z) = \sqrt{Var(Y)Var(Z)} \tag{2.11}$$

so eq.(2.10) can be written as

$$Var(Y^{'}) = (\cos^2\theta\tan^2\theta_T + \sin^2\theta - \sin(2\theta)\tan\theta_T)Var(Z) \tag{2.12}$$

When the estimated angle $\theta \to \theta_T$, the corresponding heights of the ground points will be concentrated ($Var(Y^{'}) \to 0$) to make the peak reach the maximum. So it gives us the clue to find the most accurate camera tilt angle θ (corresponding to the height distribution with the highest peak).

For the definition of Δ, if $P = \{P_G\}$, we expect $\tilde{\sigma} \propto Var(Y^{'})$ so that the size of Δ could be convergent as well. However, it is unachievable since we can only calculate $Var(Y^{'})$ in the case of $P = \{P_G, P_O\}$. Since we have assumed that the heights of P_G and P_O are independent, the variance of $Y^{'}$ will be:

$$Var(Y^{'}) = Var(Y_G^{'}) + Var(Y_O^{'}) \tag{2.13}$$

when $\theta \to \theta_T$, $Var(Y_G^{'}) \to 0$, but $Var(Y_O^{'})$ is difficult to be predicted and so is $Var(Y^{'})$. To solve this problem, we use the bin width w to define the range, which is given by:

$$\Delta = [y_{max} - nw, y_{max} + nw] \tag{2.14}$$

where n is the number of the bins. It is reasonable to use w since it only depends on the entire range of the height, which can be considered as a constant. This modification is simple but it will improve the performance significantly. Details of the parameter setting will be discussed in the part of experimental results.

2.2 Multiple ground planes detection

In the real world, an indoor scene might contain multiple planes with different shapes and orientations. In this thesis, we only concern about the planes which are parallel to the horizontal direction in the world coordinate system or have a tilt angle less than 45 degrees because those types of planes are more useful in some applications such as pedestrian tracking and moving robots navigation. Other planar objects perpendicular to the ground (Wall, Board, etc.) can be detected through a segmentation or labeling task, however, it is not in the scope of our work in this chapter.

In this section, based on the previous work, our algorithm is devoloped for the purpose of detecting multiple ground planes. Also, angle estimation is applied to refine the results.

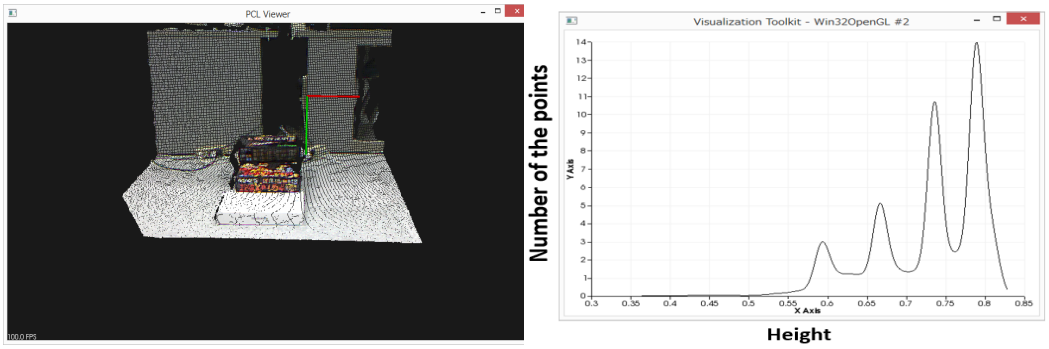
2.2.1 Parallel ground planes detection

Detecting multiple parallel ground planes can be started by observing the height distribution as well. In the case that a scene contains several parallel ground planes, since the normal vectors of them are in the same direction, the height distribution of the points after being projected onto the normal vector will show multiple peaks. As shown in Fig.2.7 (b), the four peaks correspond to four horizontal planes in the input points cloud (Fig.2.7 (a)), and the position of each peak indicates the height of each ground plane. Similarly, if the estimated angle is close to the true value, the shape around each peak is getting sharper. Meanwhile, more points belonging to each plane will gather around the position indicated by the corresponding peak. Typically, detection of multiple parallel ground planes is similar to the algorithm proposed for detecting the single

plane. The only difference is that the location and the range are not single anymore. According to this, we modify θ-projection to make it contain multiple ranges, each of which is given by

$$\Delta_i = [M_i - nw, M_i + nw] \tag{2.15}$$

where M_i is the height corresponding to the i-th peak of the distribution. Using w to define the range helps to prevent each sub-range from overlapping, which is more likely to happen when using the variance. To find each peak, we calculate each maximum of the distribution then choose several biggest ones. The points falling into the range defined by eq.(2.15) will be the hypothesis of ground points. Since the camera tilt angle remains consistent, we run the similar steps in RD procedure to find θ_{RE}, then run PD to determine the final angle θ_P and find those planes. In this time, the number of the clusters (K) of K-means used in PD will be the same as the number of the peaks. However, since there may be several small peaks existed which are caused by P_O, we use a constraint to filter them out thus making the result more reliable. The peak will be considered as valid only if its value exceeds a pre-defined threshold. The refining steps are same to that we have proposed above.

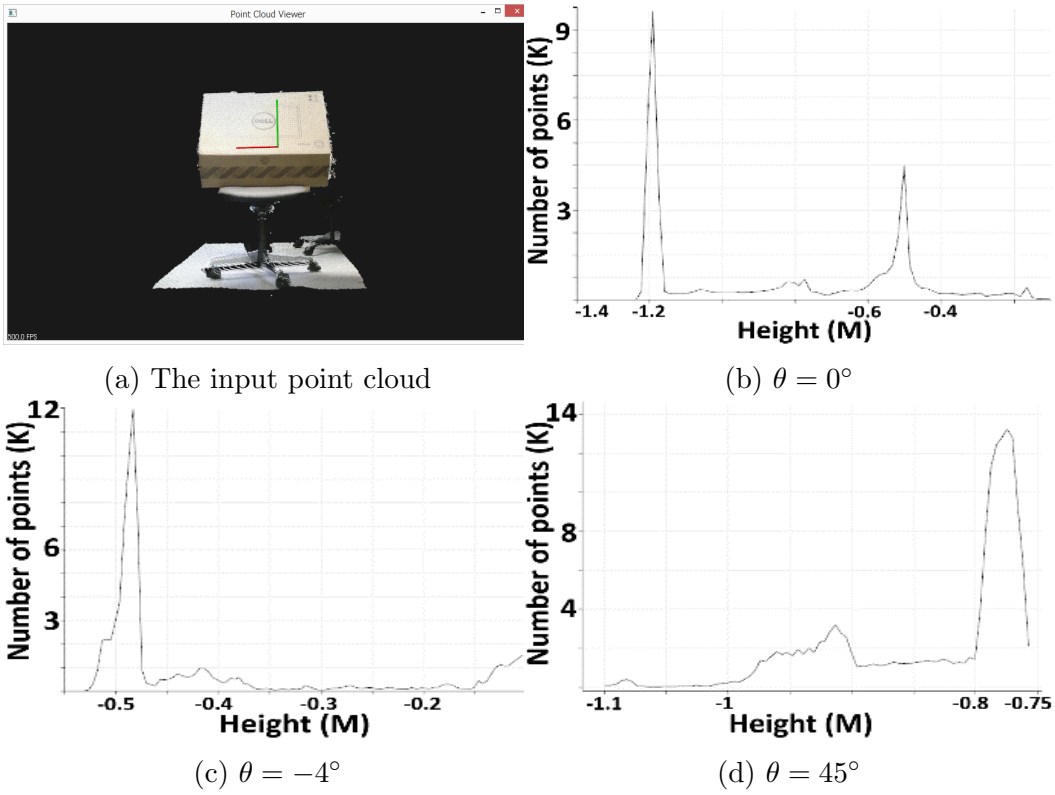


(a) The input point cloud (b) The height distribution of (a)

Figure 2.7: Figure (b) shows the height distribution of (a), in which there are four peaks. Each peak indicates the location of the corresponding plane

2.2.2 Non-parallel ground planes detection

Detecting multiple planes which are not parallel is more complicated. We first analyze the characteristics of the corresponding height distribution. For a scene containing two unparalleled grounds (each angle of which between the normal vector and the vertical



(a) The input point cloud

(b) $\theta = 0°$

(c) $\theta = -4°$

(d) $\theta = 45°$

Figure 2.8: Height distribution under different values of the estimated angle θ

direction is θ_1 and θ_2), with the estimated angle θ moving away from θ_1 but closing to θ_2, one peak is getting higher, while the other one is becoming lower, as shown in Fig.2.8

As we discussed before, position of the peak indicates where the plane exists. We record the heights when each peak reaches their maximum, then based on them, we divide the whole cloud into two parts and ensure each of them contains only one plane. Then we can use our method to detect the plane separately. In this case, camera tilt angle is difficult to estimate since we are unware of which plane is horizontal. However, if a ground plane is contained in the scene, the camera tilt angle will be equal to the angle estimated for the most bottom plane. This approach is also adapted for scene containing more than two planes. Meanwhile, angle between two neighboring planes can be estimated as well.

2.3 Experimental results

To confirm the effectiveness of the proposed method, we first discuss the influence brought by several parameters, then do some comparative experiments with RANSAC both in the scene of single-ground plane and multi-ground planes respectively. We took three datasets consisting of different scenes with variety of objects. All the datasets are taken from our laboratory under different illumination conditions. Each dataset was made up of 80 frames of point clouds. Two datasets were taken by a stationary camera with moving objects on the ground, and the other one was taken by a camera changing the tilt angle continuously. The camera we used is Kinect 2.0 and the CPU used in our work is Intel Core i7 with the frequency 3.5GHz. Additionly, we took several points clouds with multiple planes which are either parallel or unparalleled for verifying the effectiveness of our method for multiple planes detection.

Table 2.1: Details of the parameters used in our experiments.

Parameters	**Range**	**Value in our work**
Number of Bins in (2.5) (Histogram)	$[450, 550]$	512
Iteration of K-Means	None	20(max)
n in (2.14)	$[8, 12]$	10
threshold ϵ	None	2°
θ_ϵ in PD	None	5°
Threshold used in K-Means	None	0.02
Threshold for peaks	None	$20\%N(G_{input})$

2.3.1 Details of parameters

As the key component in our algorithm, θ-projection decides the final performance of the system. As discussed in section 2.1, using the bin width w is better than that using the variance σ. We first gave a list for comparing the number of points detected by using the parameter σ and w. As shown in Table 2.2, the number of the points in the range Δ defined by w reached the maximum when the estimated angle was equal to the ground truth, which made the angle calculated by eq.(2.8) more reliable in the RD procedure. In the case of using σ, the number increased with the estimated angle, which would cause a failure result.

For the parameter w, in addition to the height range of the scene, it also depends on the number of the bins constituting the whole height histogram. As shown in Table 2.1, this number was limited within a range of $[450, 550]$, and experimental results of our datasets were confirmed to be stable in this range. Results of the following experiments were obtained by using 512 bins.

After w was decided, the value of n was set in a range of [8, 12] as shown in Table 2.1. This range was decided according to the entire experimental results empirically. Figure 2.9 shows that $n = 10$ gives the best consequent. Meanwhile, results brought by other n in this range are stable as well. Using other values out of this range made the final result worse since it made $\theta_R E$ too deviated from the ground truth, then affected the final results as shown in Fig.2.10. Lots of points on the ground plane were not detected when n was under 8, while points not belonging to the ground plane were mistakenly detected when n was too large (over 12). Besides, the chosen of n and w might be changed according to the accuracy of the camera.

For K-Means used for multiply planes detection, K is determined by the number of the valid peaks whose value exceeds a pre-defined threshold as we discussed in subsection 2.2.1. In our work, this threshold was defined to be 20% of the number of G_Input according to the proportion of the smallest ground plane to the corresponding scene in our dataset.

Other parameters used in this work were also set empirically as shown in Table 2.1. The calculation of the detection accuracy for each points cloud is given by

$$Accuracy = 1 - \frac{N(MissDetected)}{N(Ground)} \tag{2.16}$$

where $N(MissDetected)$ is the number of the points wrongly detected and un-detected. $N(Ground)$ is the number of ground points manually denoted for each points cloud in our datasets.

2.3.2 Single/Multiple ground plane detection

Single ground plane detection with a stable camera

Figure 2.11, 2.12, 2.13 and Fig.2.14 illustrate the results by using our approach on three datasets. For the first two datasets created by a stationary camera, the angle was constant so that position of the ground plane would remain stable. In order to

Table 2.2: Number of detected points by using different parameters and angles

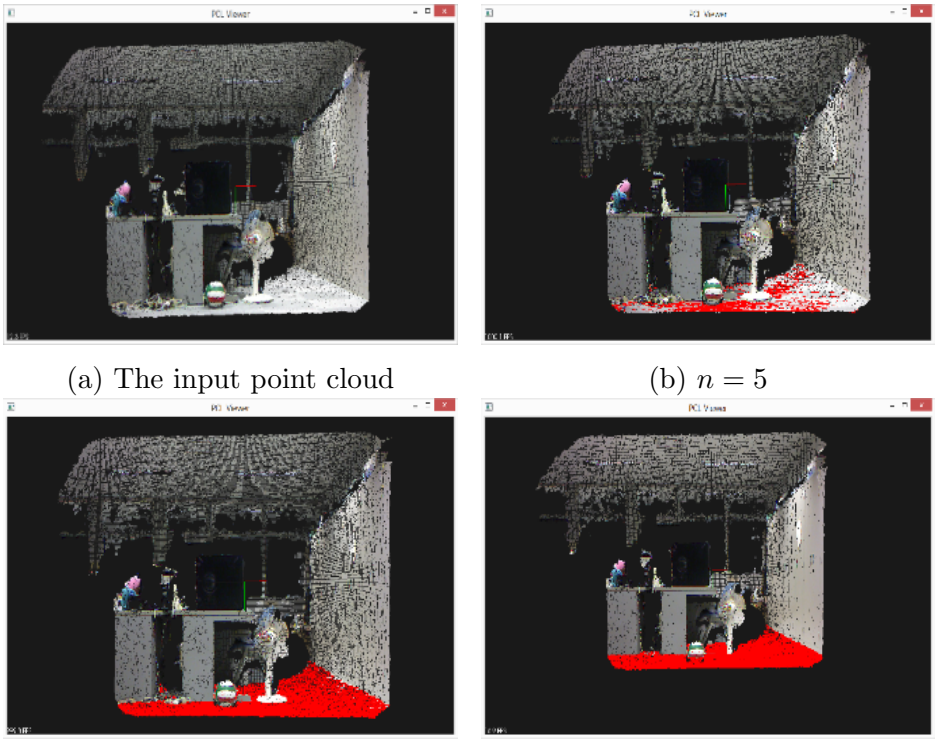
Estimated Angle	**Using w**	**Using σ**
$\theta = 0°$	25689	26843
$\theta = 1°$	26545	27065
$\theta = 2°$	26643	27125
$\theta = 3°$ (**true value**)	**26670**	27094
$\theta = 4°$	18960	**29845**

increase accuracy, after PD procedure, the mean height of the ground points calculated by K-means in the current frame will be passed to the next frame and used as an initial for training. After we got the updated mean height, (that would be the position of the ground plane), we can run K-Means directly to find the ground plane without estimating the angle since we have already known its value. As Fig.2.15 illustrates, the accuracy rate of detection is increasing gradually.

In the first dataset, only one person is standing on a narrow ground plane. We made the person stand on the different locations so that the shape of the gound plane in each frame is not same. We also added other objects to change the shape of the ground plane in each frame. In the second dataset, we make two persons moving from far to close on a wide ground plane. Meanwhile, the locations where they standed are changing. From the results it can be clearly seen that our algorithm is robust to the shape of the ground plane.

Single ground plane detection with a moving camera

We used our method on the third database created by a moving camera and the results are shown in Fig.2.14. Since the angle was changing through the whole frames, position of the ground plane was no longer stable. However, changing of the angle between two neighboring frames was very small, for saving computing time, instead of running RD procedure for the next frame, we used the angle obtained from the last frame as the roughly estimated one to perform PD procedure directly, that is, the range used in PD for the current frame was centered on the angle obtained from the last frame. Similarly, RD procedure with respect to the following frames could be omitted. The accuracy rate of detection is shown in Fig.2.15.



(a) The input point cloud (b) $n = 5$

(c) $n = 10$ (d) $n = 15$

Figure 2.9: Ground points detected (illustrated as red color) by using different n in equation (2.14)

Multiple planes detection

Method in [20] was proposed for ground plane detection by using a TOF camera. It is hard to compare with our algorithms since in this method the ground plane is detected by using several continuous frames with spatio-temporal features. It is attempted to solve the problem that when the points belonging to an obstacle (wall) is more than that belonging to the ground plane, using RANSAC will cause a failure that detecting the obstacle as the ground plane. However, this problem can be solved by our method as well. We took several scenes in which points belonging to ground plane is not the most. Results in Fig.2.16 show that by using our method the ground plane can be detected correctly. It is due to that we used the preprocessing in section 2.1 to make sure the number of points belonging to ground plane is the most. Meanwhile, since we have set the changing range of angle to be, other planes standing on the ground will not be detected.

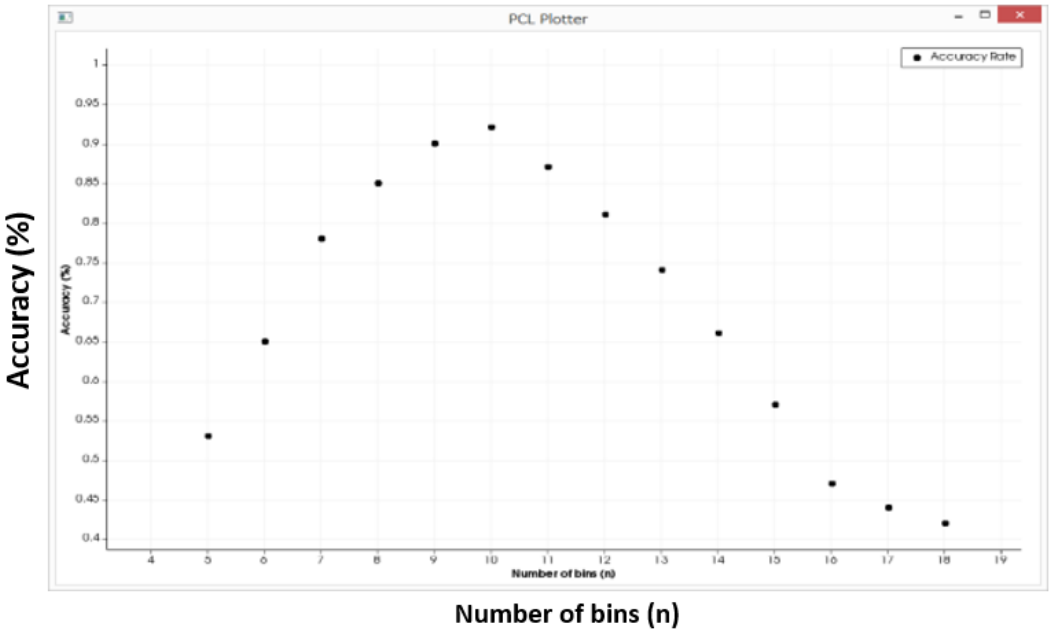
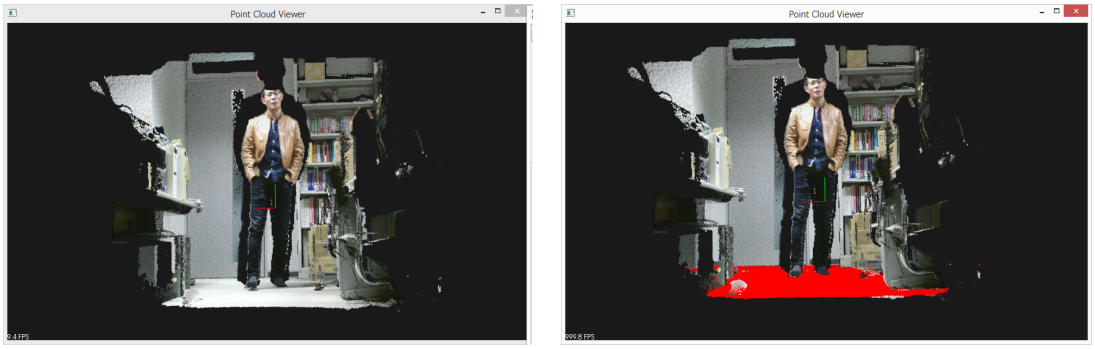


Figure 2.10: Accuracy rates brought by different n

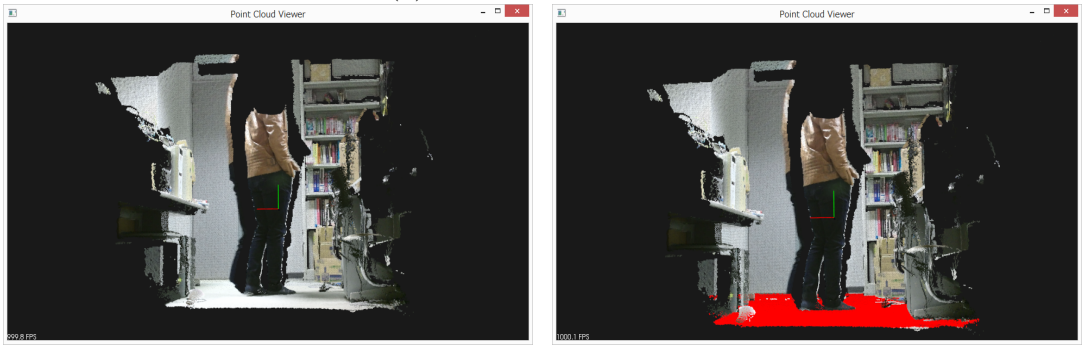
As shown in Fig.2.17, we used our method to detect multiple planes which are either parallel (Fig.2.17(a) left and middle) or unparalleled (Fig.2.17(a) right) and also did comparisons with RANSAC. Our algorithm could detect all planes successfully (Fig.2.17 (b)), while RANSAC could only detect the largest one as shown in Fig.2.17 (c). It was caused by that the model it used was designed to detect the single ground plane, making it not suitable for the scene containing multiple ground planes.

To confirm the effectiveness of our method, we also compared with CC-RANSAC in [18]. We used a simple synthetic stairs scene to show the limitation of CC-RANSAC. Input points clouds and the results are shown in Fig.2.18. For the scene on the right part of Fig.2.18 (a), results brought by the two methods are much closed. However, for the scene on the left part (stairs), it is clear that CC-RANSAC caused a worse result. Since CC-RANSAC uses the largest connected inlier, some part of riser connected to tread was mistakenly detected. While by using our method, each position of tread can be found explicitly, so this problem can be avoided. Moreover, CC-RANSAC didnt solve the speed problems existing in the RANSAC method, so that the computational cost was still very large.

Finally, we have compared our method with the modified Hough transform in [30]

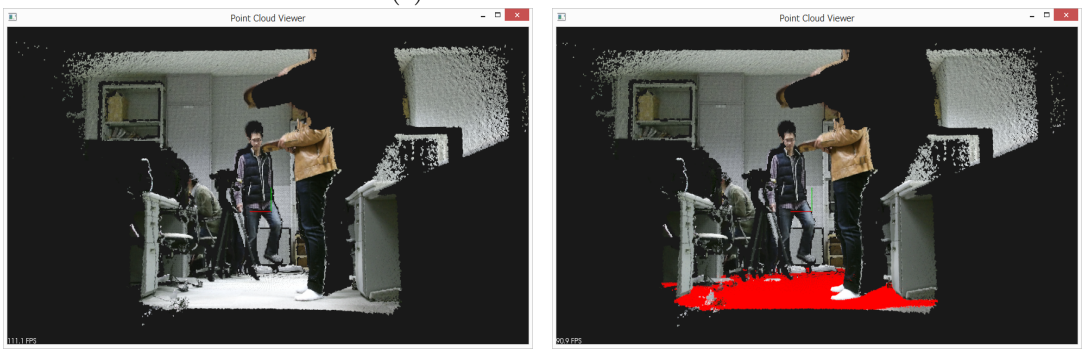


(a) frame 11 of dataset1



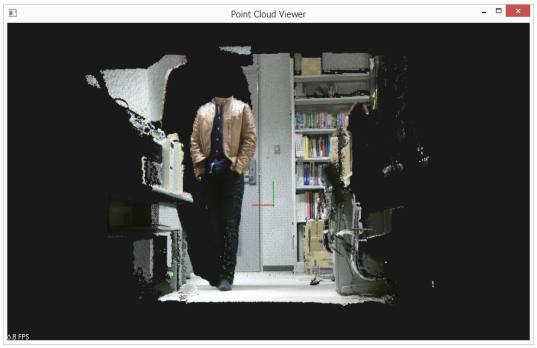
(b) frame 21 of dataset1

(c) frame 11 of dataset2

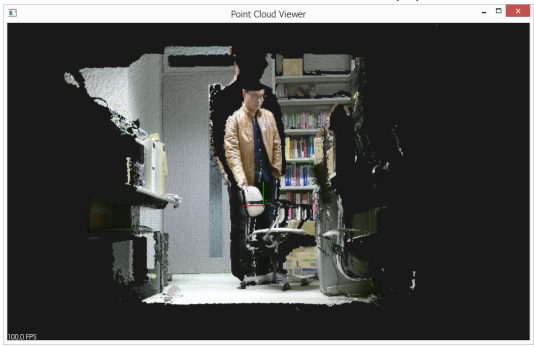


(d) frame 21 of dataset2

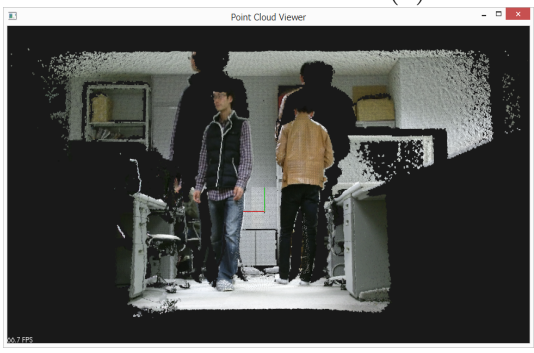
Figure 2.11: Result of the fixed camera captured dataset



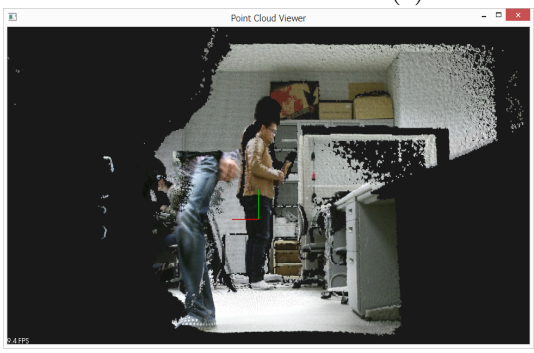
(a) frame 15 of dataset1



(b) frame 27 of dataset1

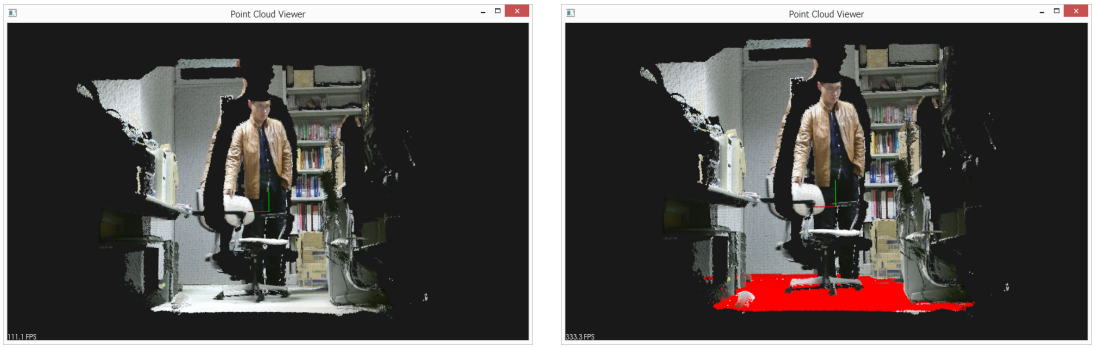


(c) frame 15 of dataset2

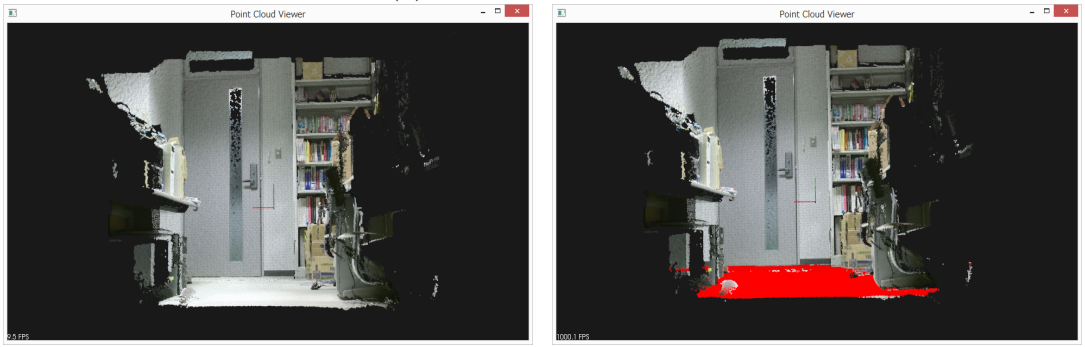


(d) frame 27 of dataset2

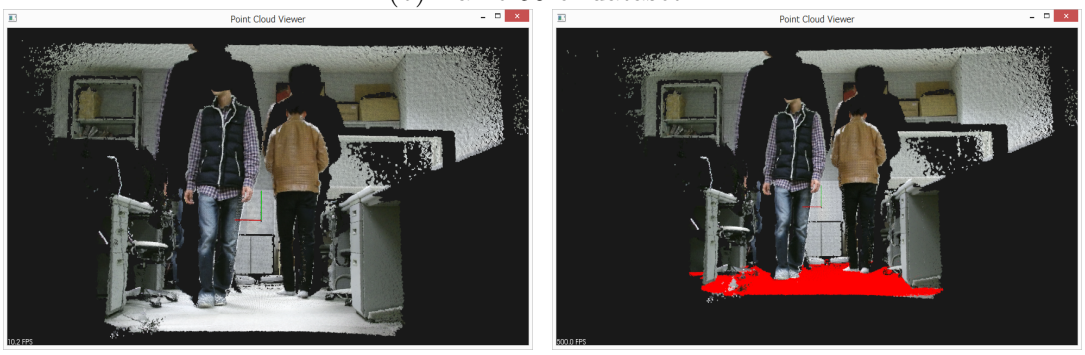
Figure 2.12: Result of the fixed camera captured dataset



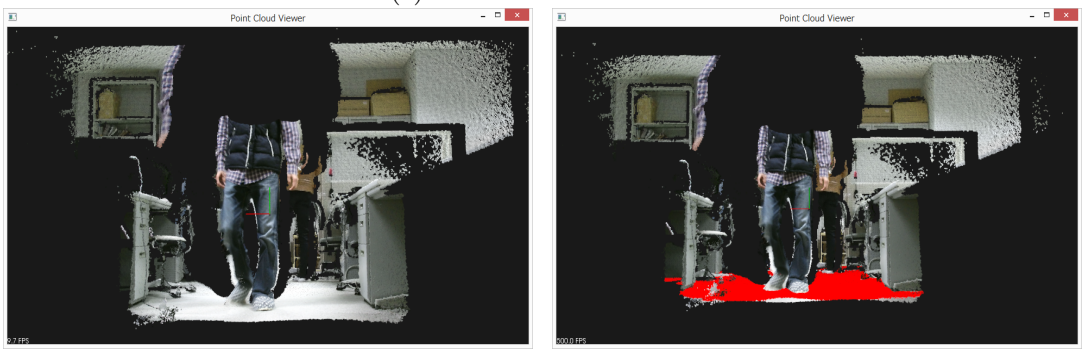
(a) frame 35 of dataset1



(b) frame 50 of dataset1



(c) frame 35 of dataset2



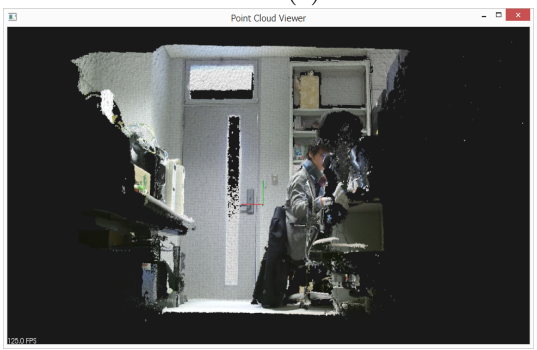
(d) frame 50 of dataset2

Figure 2.13: Result of the fixed camera captured dataset

(a) frame 30 of dataset3 when $\theta_E = 3°$

(b) frame 33 of dataset3 when $\theta_E = 0°$

(c) frame 36 of dataset3 when $\theta_E = -3°$



(d) frame 39 of dataset3 when $\theta_E = -5°$

Figure 2.14: Result of the moving camera captured dataset

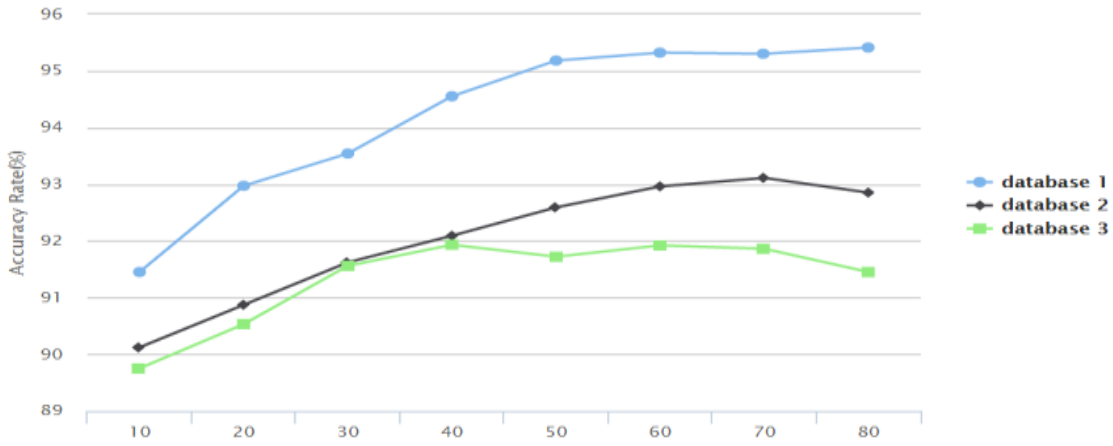


Figure 2.15: Accuracy rates of the performance on the three datasets

for multi-planes detection. For the stairs scene, with our method, steps are detected correctly. However, planes detected by RHT consist of not only treads but also risers, which is not the purpose in our research. We added a constraint that planes having horizontal normal vectors were not considered and the result is shown in Fig.2.18 (d). Table 2.3 gives the accuracy and processing speed of each method. For RHT, it used the plane patch (triple points) for voting. Since each plane patch was randomly selected, it reduced the probability that all the patches were selected from the correct plane. This uncertainty makes the votes in the parameter space scattered. For this reason, with a low number of iterations, a result with low accuracy was obtained as shown in Fig.2.18 (c), even though the speed was close to our method. To reduce the uncertainty brought by the plane patch, increasing the number of iterations can make the votes in the parameter space more reliable, and the result with an improved accuracy is shown in Fig.2.18 (d), but it also increased the computational cost, which is shown in Table 2.3. For our method, the parameter space used for determining the camera tilt angle and the ground plane is the height distribution, the dimension of which (1D) is smaller than that of RHT (3D), so the processing time can be reduced significantly. In PD procedure, since the angle range $[\theta_{RE} - \theta_\epsilon, \theta_{RE} + \theta_\epsilon]$is very small, the tilt angle can be quickly estimated with less than 10 iterations. These factors made our method gain a low computational cost. Meanwhile, since the height distribution was calculated from all the points, the result of voting (for camera tilt angle) is more reliable than randomly selecting triple points as used in RHT. Thanks to K-Means used in PD procedure, it

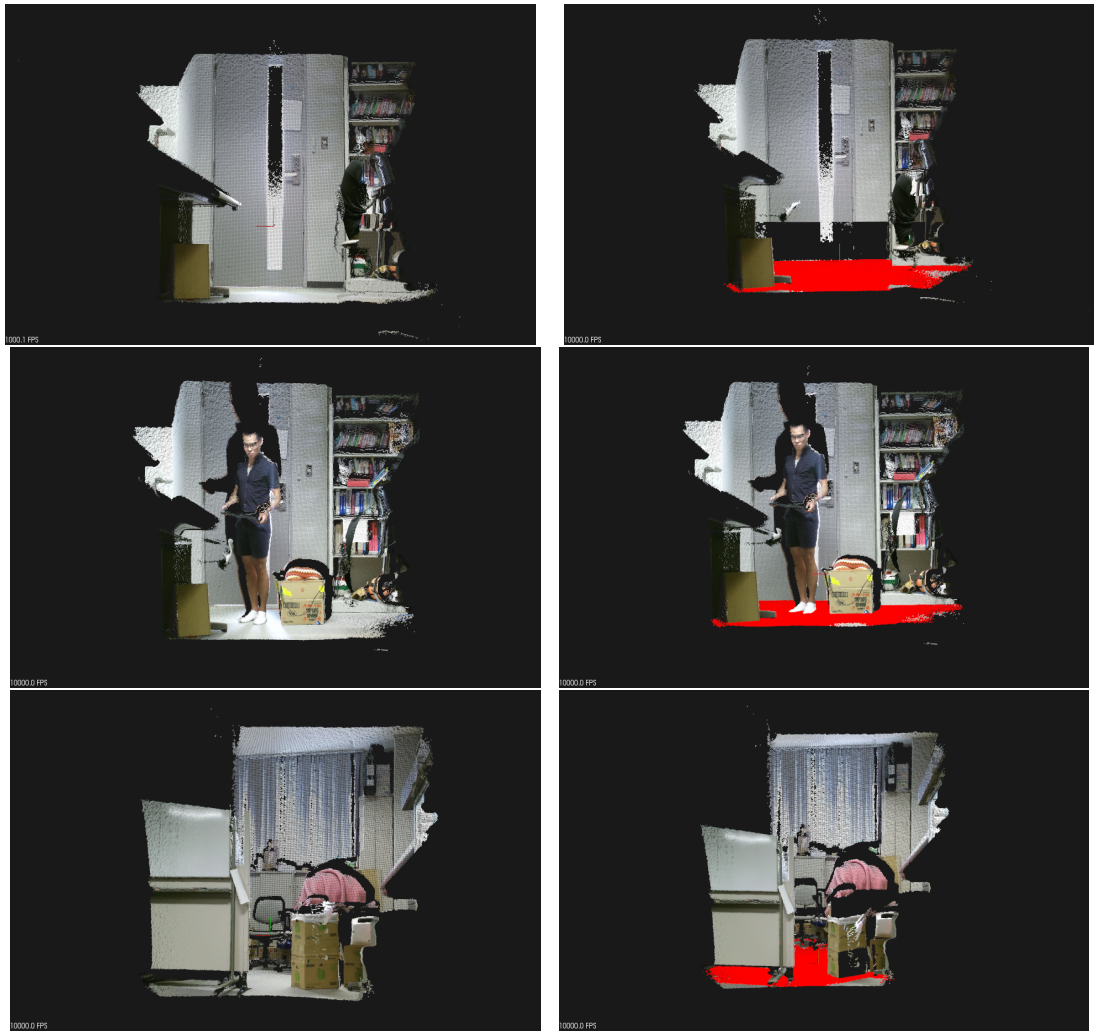
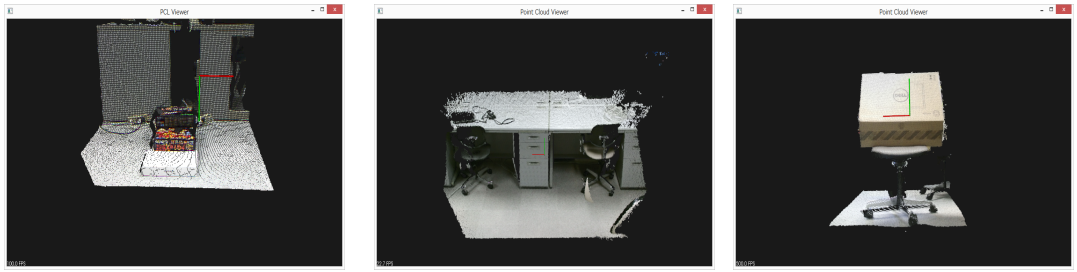


Figure 2.16: Results of detecting ground plane that is not the largest part in the scene

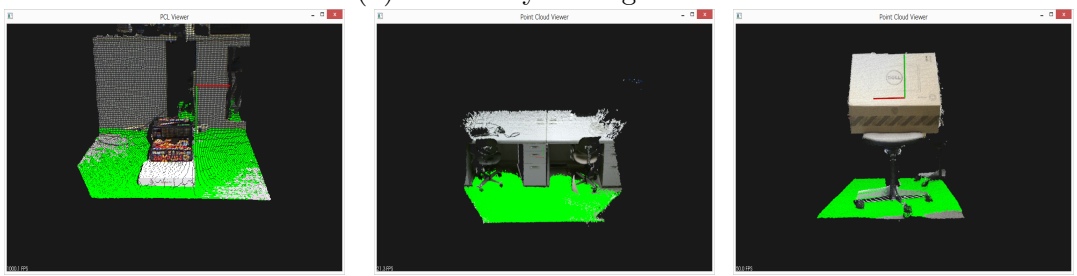
made the determination for the camera tilt angle more reliable, so that improved the accuracy of the ground detection consequently. According to these reasons, our method gained a better performance.

For all the datasets, the bias between the estimated angle and the ground truth is less than 2°.



(a) Input point cloud containing multiple planes

(b) Results by our algorithm



(c) Results by RANSAC

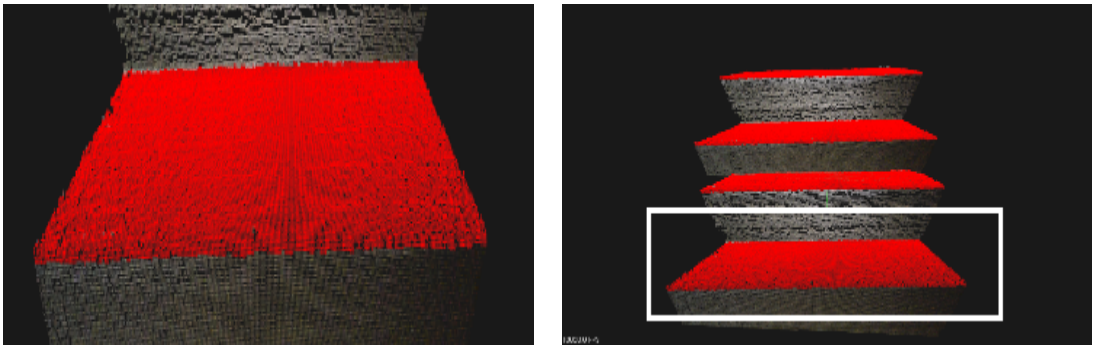
Figure 2.17: The figure shows the comparative results gained from using our algorithm and RANSAC

2.4 Conclusions and discussions

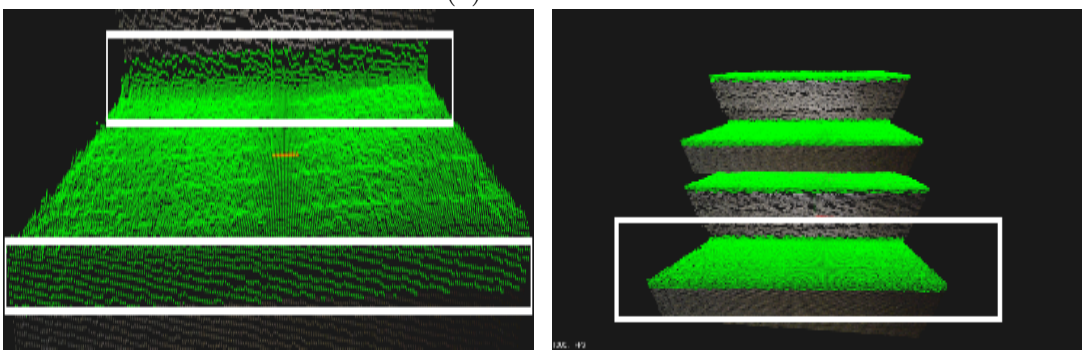
In this Chapter, we have presented an innovative algorithm named for detecting single/multiple ground planes and estimating the tilt angle of the camera. The algorithm incorporates several novel ideas:

1. We have used Kernel Density Estimator for selecting the ground plane reliably under a given tilt angle;

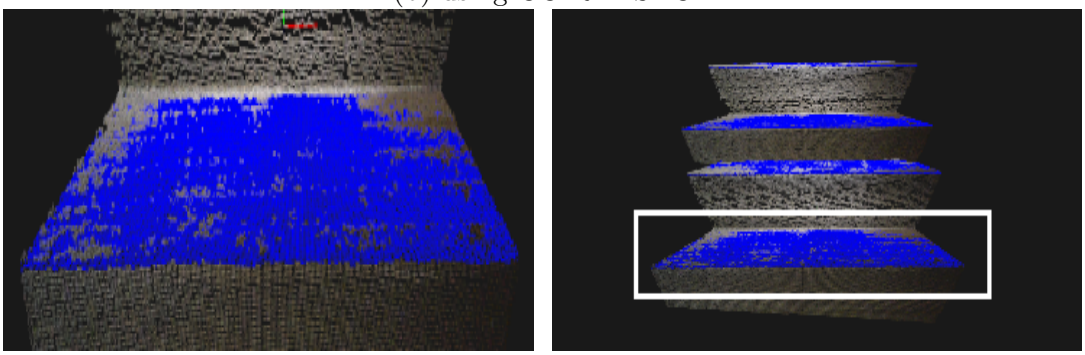
2. We have used a height distribution based approach to estimate the camera tilt angle, and used it to refine the accuracy of the ground plane detection;



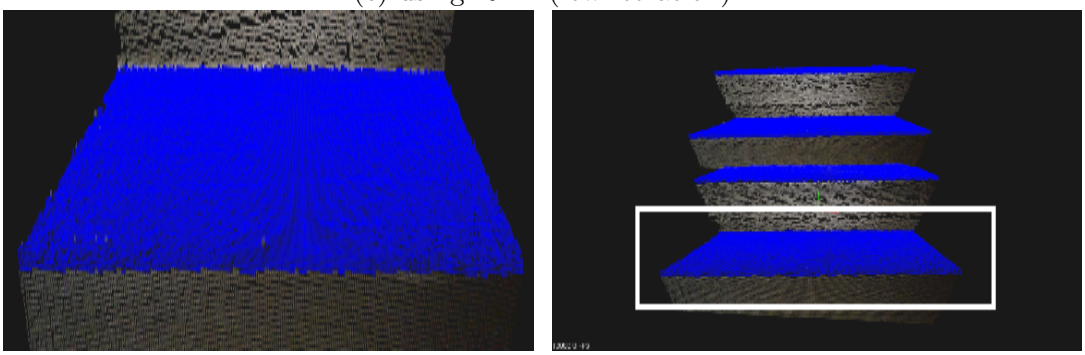
(a) our method



(b) using CC-RANSAC



(c) using RHT (low iteration)



(d) using RHT (high iteration)

Figure 2.18: Results of multiple planes detection and the comparison with CC-RANSAC and RHT. Figures on the left are results obtained by using different methods. Details in the white frame are shown in the figures on the left side.

3. We have enabled the detection of multiple ground planes (parallel or unparalleled).

Extensive experiments have been implemented and the results confirmed the effectiveness of our approach. However, since we assumed the camera roll angle to be zero, this algorithm is only available for detecting the plane of which normal is perpendicular to the horizontal axis. In the future, we will focus on making it available for the non-zero roll angle camera, and apply it to people detection and object tracking.

Table 2.3: Accuracy and computational cost (processing speed for each frame) of each method

Method	**Iterations**	**Accuracy**	**Processing time**
CC-RANSAC	300	85%	800ms
RHT	300	92%	500ms
RHT	150	84%	180ms
Our method	150	**93**%	**160ms**

3

3D Indoor Scene Labeling with Markov Random Field

In this chapter, a Bayesian Framework is proposed to label the objects in the 3D scene with several meaningful categories. In order to achieve the robust labeling performance, it is necessary to develop a structural feature vector to describe the characteristics of the objects in the scene. Moreover, the spatial relevance between different categories should be captured. Apparently, the label of the object has a strong dependency on the neighbors. When the features are not sufficiently discriminative to predict the labels, exploiting the spatial relationship can significantly improve the performance. Meanwhile, it is helpful to reach a large reduction in computation and false positives since the range of hypothesis can be limited.

The labeling tasks can be naturally posed as energy minimization problems, where the energy comprises a data cost term and a smoothness term. The data cost term expresses the optimal solution for assigning the label based on the feature. The smoothness energy is derived from our prior knowledge about the spatial relevance. According to this consideration, Markov Random Field (MRF) is regarded to be suitable to solve the labeling problem because of its ability to represent certain dependencies between different labels. On the other hand, since Markov Random Field is a graphical model of the joint probability distribution, labeling can be easily achieved by minimizing the cost energy.

In order to achieve a robust labeling performance, the following key ideas are used in this thesis:

1. Using eigenvector decomposition and sub-space combination to capture the structural feature of the objects.
2. Applying a special likelihood density to calculate the data cost energy.
3. Designing a 6-connected pair-wise model that accommodates the relationship of the 3D location.
4. Using this model to define the smoothness cost energy.
5. Designing a method to solve the problems caused by dividing the 3D scene with the same size.

The content of this chapter is organized as follows:

1. In section 3.1, we give an overview of the proposed 3D indoor scene labeling algorithm.
2. In section 3.2, method of developing the feature vector used in this research is proposed.
3. In section 3.3 and section 3.4, we give the details of the proposed function of calculating the data cost energy and the smoothness cost energy respectively.
4. In section 3.5, we give the details of using Loopy Belief Propagation to find the optimal labeling solution.
5. In section 3.6, we present the experimental results comparing with other approaches.
6. In section 3.7, we make a short conclusion and discussion.

3.1 Overview of our approach

Markov Random Field

A Markov Random Field (MRF) is a graphical model used for describing the joint probability distribution of an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_N\}$ is the set of nodes, each of which is associated with a random variable, u_j, for $j = 1, \ldots, N$.

This means that the state of the current node u_j only depends on its neighbors, denoted as E_i, which is the set of nodes to which i is adjacent; i.e., $j \in E_i$ if and only if $(i, j) \in E$. The Markov Random field satisfies $p(u_i|\{u_j\}_{j\in V}) = p(u_i|\{u_j\}_{j\in E_i})$. For this property, Markov Random Field is usually used for labeling in 2D images. As

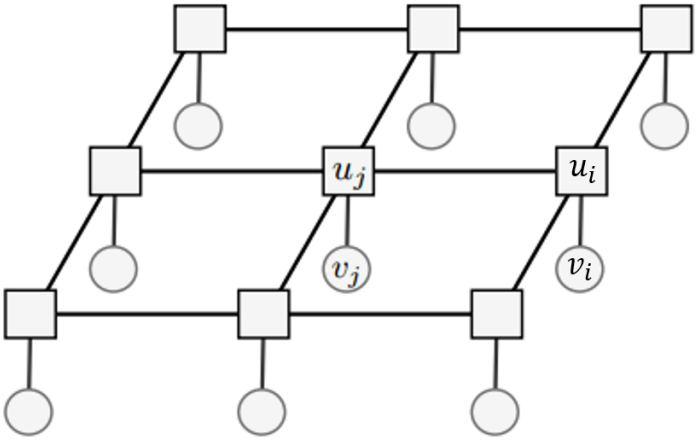


Figure 3.1: Graphical model used for 2D image, where each pixel is represented as a node.

shown in Fig.3.1, each pixel is represented as a node with 4-connected neighborhoods in the graph $G = (V, E)$. v_j is the observation of the node, and u_j is the label where $u_j \in \{l_1, \ldots, l_M\}$. Accordingly, the energy function is given by

$$E(\mathbf{u}) = \sum_{i\in V} D(u_i) + \sum_{(i,j)\in E} S(u_i, u_j). \tag{3.1}$$

The unary term $D(\cdot)$ penalizes the cost energy for assigning the label u_j to the node j conditioned on the observation v_j. This model assumes conditional independence of observations. The term $S(\cdot)$ provides a definition of smoothness, penalizing changes in u between pixels and their neighbors. The objective is to find the labels $\mathbf{u}$ that minimize $E(\mathbf{u})$.

Markov Random Field can be applied for labeling the 3D scene. However, different to 2D image, since there are always numerous points in the point cloud, representing each point as a node will bring about lots of computational cost. Using the "cube" that contains a number of points is suitable to solve this problem. In addition to that, rich information can be exploited from each cube, which is supposed to improve the

performance. The overview of our labeling framework based on Markov Random Field is given below.

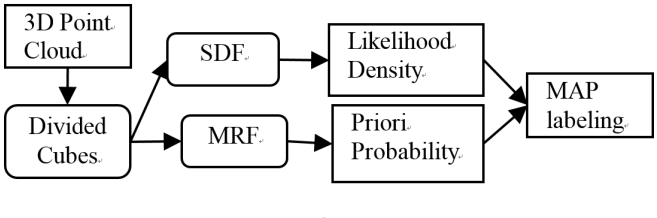


Figure 3.2: The overview of our proposed method

The proposed framework

The proposed labeling framework is shown in Fig.3.2 which is based on MRF. The labeling problem is solved by Maximizing a Posterior (MAP), which is given by

$$l^* = \arg\max_l p(l \mid \mathbf{f}) \propto p(\mathbf{f} \mid l)p(l), \tag{3.2}$$

where l is the label and $\mathbf{f}$ is the feature vector. In our method, first, the whole 3D point cloud is divided into a certain number of cubes, each cube is represented as a node in the Markov Random Field and is supposed to be labeled as $l = \{l_1, l_2, \ldots, l_M\}$, which is a random variable. For each cube, in order to calculate the conditional likelihood density $p(\mathbf{f} \mid l)$, which represents the probability distribution of the observation f based on each label l, we develop a new structural feature vector called Spatial Distribution Feature (SDF) by using eigenvector combination. Details of this feature and the conditional likelihood density $p(\mathbf{f} \mid l)$ will be discussed in the next sections. Then, a pair-wise model representing the label relationships between neighboring cubes is designed for calculating the prior probability $p(l)$. The MAP estimation is equivalent to minimizing the cost energy, which is obtained by using negative logarithm for both side of equation 3.2. The cost energy is given by

$$E_{cost}(l) = E_{datacost}(\mathbf{f} \mid l) + E_{smoothness}(l), \tag{3.3}$$

where the data cost energy is

$$E_{datacost}(l) = -\log\{p(\mathbf{f}|l)\}, \tag{3.4}$$

and the smoothness energy is

$$E_{smoothness}(l) = -\log\{p(l)\}. \tag{3.5}$$

We use Loopy Belief Propagation to solve the inference problem, that is, to find the most appropriate l that minimizes the cost energy for each cube. Details of the inference will be discussed in section 3.5. Categories of all objects set for labeling consist of *Roof*, *Ground*, *Wall* and *Objects* (represented as R, G, W and O in the next) so the label of each cube is expressed by $l \in \{l_R, l_W, l_O, l_G\}$

3.2 Spatial Distribution Feature (SDF)

The fast development of depth camera makes it available to exploit spatial features from the 3D scene. Height distribution is a traditional feature to describe how the objects distribute along the vertical direction. The ability of this feature makes itself strong to discriminate the objects which have a specific location such as ground plane and roof. However, in this work, after dividing the point cloud, many cubes with different categories usually have the same height distribution, in which case the height distribution becomes weak to distinguish. For those cubes, since their positions cannot not be overlapped, the spatial feature of them can be enhanced by considering the distributions along the other directions. For example, even though the distributions of the vertical direction (Y axis) and the horizontal direction (X axis) of a pair of cubes are similar to each other, distribution of the depth direction (Z axis) is definitely different. Therefore, the spatial feature should be developed from all the three distributions.

The spatial features of an object can be represented by the distributions along X, Y, and Z axis in the world coordinate system, in the form of three histograms ($\mathbf{r}_X$, $\mathbf{r}_Y$ and $\mathbf{r}_Z$,) with the same number of bins. Typically, we can simply combine those histograms into a high dimensional vector and use it to describe the entire spatial feature. However, this method is not appropriate because it ignores the dependencies between the histograms. Moreover, the high dimension of this feature vector will bring about numerous computational cost.

Eigenvector decomposition makes it available to solve these problems. The eigenvector of a matrix is a vector mapped to a scaled version of itself, and the scale depends on the corresponding eigenvalue. So a matrix can be characterized as its eigenvectors

and eigenvalues. According to these properties, in this chapter, we use eigenvector decomposition and subspace combination to generate a feature vector for the purpose of describing the characteristic of entire spatial distribution. Details of generating this feature are given below:

Step (1): For each cube, histograms of the 3D points are calculated with respect to three coordinate directions, which are given by:

$$\mathbf{r}_X = [r_{x_1}, \cdots, r_{x_k}, A_x]^{\mathrm{T}}, \tag{3.6}$$

$$\mathbf{r}_Y = [r_{y_1}, \cdots, r_{y_k}, A_y]^{\mathrm{T}}, \tag{3.7}$$

$$\mathbf{r}_Z = [r_{z_1}, \cdots, r_{z_k}, A_z]^{\mathrm{T}}, \tag{3.8}$$

where r_{x_i},r_{y_i},r_{z_i} are the numbers of the points falling into the i-th bin. A_x,A_y,A_z are the coordinates of the centroid point of each cube. These vectors are combined to form a matrix $\mathbf{R}$=[$\mathbf{r}_X$,$\mathbf{r}_Y$,$\mathbf{r}_Z$].

Step (2): A symmetric $(k+1)\times(k+1)$ square is obtained by

$$\mathbf{S} = \mathbf{R}\mathbf{R}^{\mathrm{T}}. \tag{3.9}$$

After that, we decompose S as follows by caluculating its eigenvalues and eigenvectors.

$$\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathrm{T}}, \tag{3.10}$$

where

$$\mathbf{\Lambda} = diag[\lambda_1, \lambda_2, \cdots, \lambda_{k+1}] \tag{3.11}$$

$$\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_{k+1}], \tag{3.12}$$

where λ_i is the i-th eigenvalue of $\mathbf{S}$ and $\mathbf{v}_i$ is the corresponding eigenvector. Without losing generality, we assume that $\lambda_1 \geq \lambda_2 \geq, \cdots, \geq \lambda_{k+1}$.

Since $\mathbf{S}$ is a symmetrical matrix, $\mathbf{\Lambda}$ and $\mathbf{V}$ can be calculated with Singular Value Decomposition (SVD) algorithm. Meanwhile, all the eigenvectors $\{\mathbf{v}_i\}$ are linearly independent.

According to what we discussed above, eigenvector $\mathbf{v}_i$ corresponding to a large eigenvalue shows the principal characteristic of matrix $\mathbf{S}$, while those corresponding to small eigenvalues can be ignored. For this consideration, we can choose only M numbers of eigenvectors corresponding to the largest eigenvalues to represent the spatial feature. M is the minimun value satisfying the condition:

$$\frac{\sum_{i=1}^{M} \lambda_i}{\sum_{j=1}^{k+1} \lambda_j} > \eta, \tag{3.13}$$

where η is the accumulated proportion. Its value will be set empirically based on numerous experimental results.

Step (3): After M is decided, we use the linear combination of these eigenvectors with their corresponding eigenvalues to form the 3D spatial distribution feature vector $\mathbf{f}$, which is given by

$$\mathbf{f} = \sum_{i=1}^{M} \lambda_i \mathbf{v}_i. \tag{3.14}$$

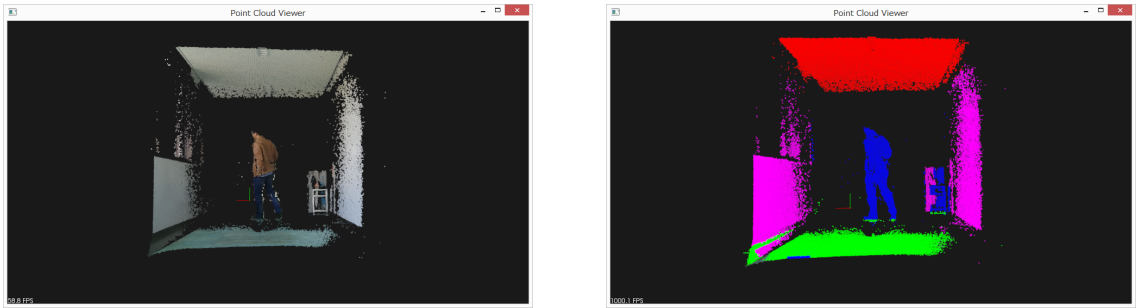
$\mathbf{f}$ is a feature vector that describes the principal characteristics of spatial distribution of the points in a cube, while discarding some useless information brought by the eigenvectors with small eigenvalues, which can be viewed as noise. Meanwhile, since it is formed by linearly combining several orthogonality eigenvectors, the information indicated by each eigenvector is independent. According to this, $\mathbf{f}$ is named as Spatial Distribution Feature (SDF). We use this feature vector to calculate the likelihood density conditioned on each label. Details will be discussed in the next section.

3.3 Conditional likelihood density

The Spatial Distribution Feature captures the distribution of 3D points in any given scene in terms of linear combination of eigenvectors. In order to associate each 3D cube with structural labels, we compute likelihood density of the proposed feature vector conditioned on the specific structural label respectively. First, a small number of point cloud from the entire dataset are randomly chosen, in which the divided cubes are hand-labeled as $l \in \{l_R, l_W, l_O, l_G\}$. Since the coordinate information of each point might be negative (horizontal and vertical) in the point cloud, before the experiments,

we normalize the coordinate range of 3D point cloud into the interval [0,1] to facilitate computation. For each cube hand-labeled as l, we compute $\mathbf{r}_X$, $\mathbf{r}_Y$ and $\mathbf{r}_Z$ according to eq.(3.6)-(3.8), then calculate the mean vectors $\tilde{\mathbf{r}}_X$, $\tilde{\mathbf{r}}_Y$, $\tilde{\mathbf{r}}_Z$ respectively, which indicate the average distributions among X, Y, and Z axis, and three covariance matrix $\mathbf{\Sigma}_X$, $\mathbf{\Sigma}_Y$ and $\mathbf{\Sigma}_Z$, which are used to calculate the conditional likelihood density. In this chapter, we use Greek letter $\mathbf{\Sigma}$ to represent the covariance matrix[79]. Each histogram is divided into four bins and the range of each bin are shown in table 3.1, which is decided empirically based on numerous experiments. This can be explained from an intuitive consideration: (i), cubes labeled as R and G distribute equably along horizontal and depth direction, and have a high and low average height, respectively; (ii), points in the O cube concentrate at the center of the horizontal direction and the average height of them is low; (iii), objects labeled with W spread through vertical and depth direction. Using this boundaries will force the points to fall into the most appropriate bins, making the histograms discriminative for each label.

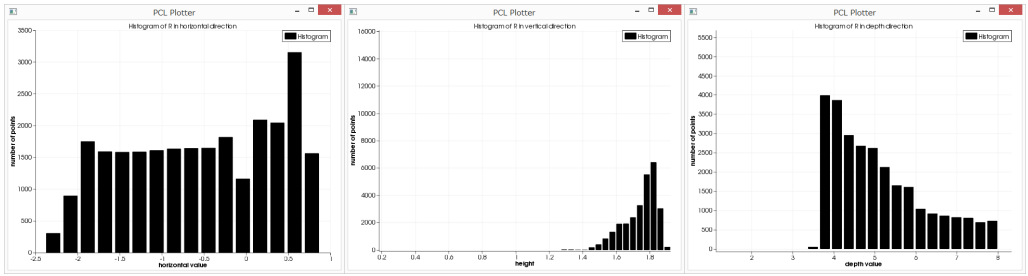
As we discussed before, assigning the cube with a label is more dependent on the hierarchy of the three histograms. Since the datacost energy measures the cost for labeling the object based on the observation, for each label, we can first calculate the cost brought by each histogram, then use a linear combination to show the entire cost energy. Hierarchy of the histograms for the corresponding label is expressed in terms of the weight. Apparently datacost energy is more dependent on the histogram corresponding to a high weight value.



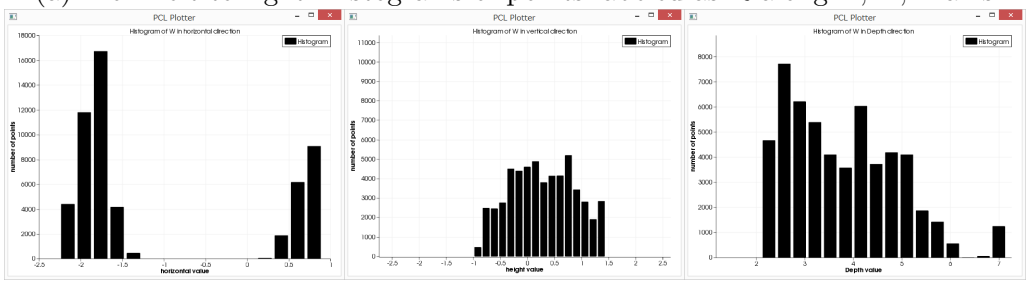
(a) Input Point Cloud (b) Labeling result

Figure 3.3: This figure shows the input Point Cloud and the labeling result

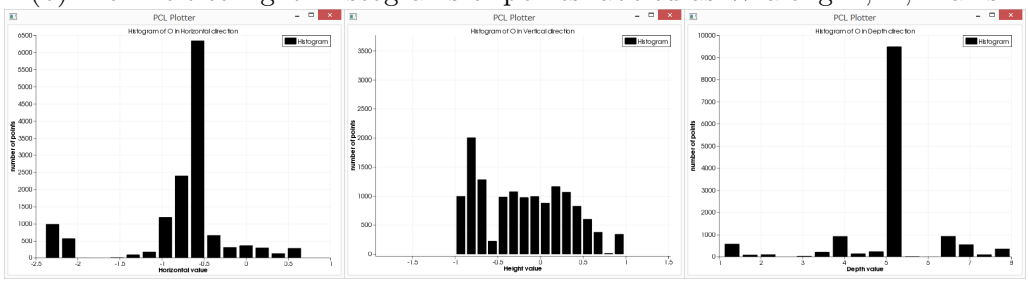
In order to design the conditional probability density to calculate this datacost energy, we first consider the contributions from the coordinate histograms $\mathbf{r}_X$, $\mathbf{r}_Y$ and



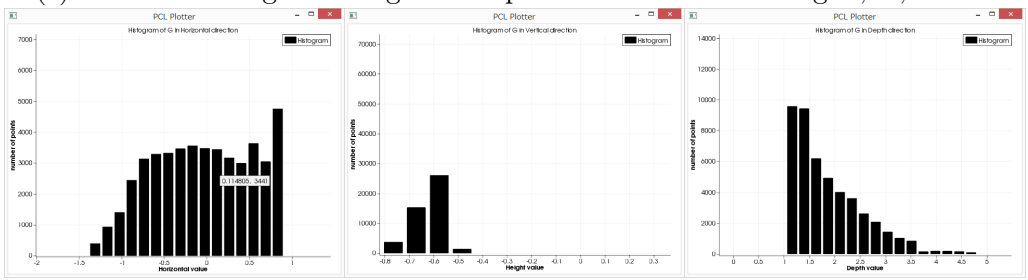
(a) From left to right: histograms of points labeled as R along X, Y, Z axis



(b) From left to right: histograms of points labeled as W along X, Y, Z axis



(c) From left to right: histograms of points labeled as O along X, Y, Z axis



(d) From left to right: histograms of points labeled as G along X, Y, Z axis

Figure 3.4: Histograms along three coordinate directions for points labeled as R, W, O and G. Figures from left to right show the histograms along Horizontal, Vertical and Depth direction. In each histogram, the vertical axis denotes the number of points, and the horizontal axis denotes the corresponding coordinate value

Table 3.1: Parameters used in our research

Cube Numbers	L		H	D
	16		16	20
Defination of α_i for each label		α_1	α_2	α_3
	R	0.1	0.8	0.1
	W	0.5	0.2	0.3
	O	0.3	0.3	0.4
	G	0.1	0.7	0.2
Histogram Boundary	Bin 1	Bin 2	Bin 3	Bin 4
X	0-0.2	0.2-0.4	0.4-0.8	0.8-1
Y	0-0.2	0.2-0.5	0.5-0.8	0.8-1
Z	0-0.4	0.4-0.5	0.5-0.7	0.7-1

$\mathbf{r}_Z$ for each label l. Figure 3.3 shows an input point cloud and the result that has been labeled as R, W, O and G by using our approach (illustrated by red, purple, blue and green color respectively). Figure 3.4 shows the histograms of the points belonging to each label, along three coordinate directions. From Fig.3.4 it is intuitive that (i), likelihood conditioned on R and G is primarily depended on $\mathbf{r}_Y$, regardless of $\mathbf{r}_X$ and $\mathbf{r}_Z$; (ii), likelihood conditioned on W is primarily depended on $\mathbf{r}_X$ or $\mathbf{r}_Z$, with less contribution from $\mathbf{r}_Y$. (iii), likelihood conditioned on O takes supports from all the three distributions. According to this, we use three parameters to define the hierarchy of each histogram, which is given by

$$\sum_{i=1}^{3} \alpha_i = 1; \quad 0 < \alpha_i < 1 \tag{3.15}$$

to specify the contributions for each label. The value of α_i for each label is decided empirically according to numerous experiments as shown in table 3.1.

After the decision of the weights, since we aim to use a linear combination of energy to represent the datacost energy, the corresponding conditional likelihood density in this work is designed as

$$p(\mathbf{f} \mid l) = \prod_{i=1}^{3} \frac{1}{Z_i} \exp\left\{-\frac{\alpha_i}{2}(\mathbf{f} - \tilde{\mathbf{r}}_i)^{\mathrm{T}} {\boldsymbol{\Sigma}_i}^{-1} (\mathbf{f} - \tilde{\mathbf{r}}_i)\right\} \tag{3.16}$$

where $\tilde{\mathbf{r}}_1 = \tilde{\mathbf{r}}_X$, $\tilde{\mathbf{r}}_2 = \tilde{\mathbf{r}}_Y$ and $\tilde{\mathbf{r}}_3 = \tilde{\mathbf{r}}_Z$ corresponding to the label l. $\mathbf{\Sigma}_i$ is a $(k+1)\times(k+1)$ covariance matrix and $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_X$, $\mathbf{\Sigma}_2 = \mathbf{\Sigma}_Y$, $\mathbf{\Sigma}_3 = \mathbf{\Sigma}_Z$. Z_i is the normalizing constant. Then, the energy of Datacost for each label $l \in \{l_R, l_W, l_O, l_G\}$ is represented as

$$\begin{aligned} Datacost(\mathbf{f} \mid l) &= -\log\{p(\mathbf{f} \mid l)\} && (3.17)\\ &= \sum_{i=1}^{3} \alpha_i (\mathbf{f} - \tilde{\mathbf{r}}_i)^{\mathrm{T}} {\mathbf{\Sigma}_i}^{-1} (\mathbf{f} - \tilde{\mathbf{r}}_i) + \sum_{i=1}^{3} \log Z_i && (3.18)\\ &= \sum_{i=1}^{3} \alpha_i Datacost_i(\mathbf{f} \mid l) + Const && (3.19) \end{aligned}$$

Alternatively, we can design the conditional likelihood density as Gaussian Mixture Density. However, since GMD is a combination of several Gaussian density, calculation of the logarithm is very difficult and costs numerous time. Moreover, training for the parameters also costs a lot of time. Our model proposed in this paper consists of only 3 components, each of which corresponds to the coordinate distribution, with a weight α_i to stress the contribution from each single distribution. It is reasonable and very simple to implement.

After we define the conditional likelihood density of the feature on each label, we need to use the priory probability of l to calculate the joint probability $p(\mathbf{f}, l)$. Details of the priory probability $p(l)$ will be discussed in the next section.

3.4 Priori probability

In general, the definition of priori joint probability is difficult to realize due to the large computation. However, since l is a random variable and depends on its neighbors, we can represent l as the hidden nodes of Markov Random Field, which makes the definition of priory joint probability become tractable. In this section, the priori probability of label l is defined as a Gibbs distribution according to the Hammersley-Clifford theorem, which is given by

$$p(l) \propto e^{-E_{smoothness}(l_i, l_j)} \tag{3.20}$$

where $E_{smoothness}(\cdot)$ is the smoothness energy. Since the identification of a cube depends on its neighboring cubes, we use a particularly designed pairwise smoothness term between neighboring cubes to model the cost function.

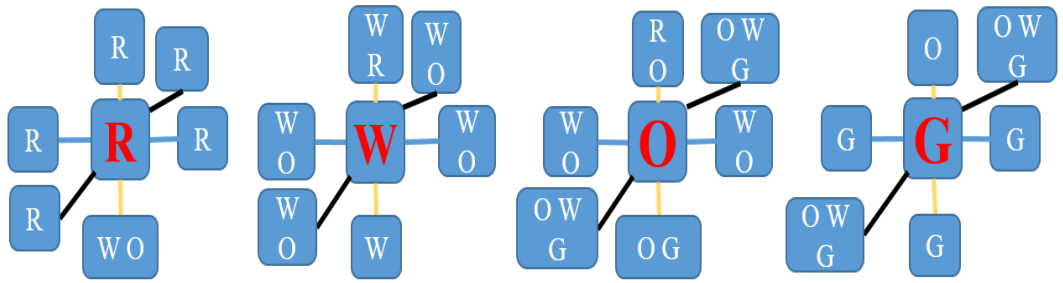


Figure 3.5: 6-connected pair-wise model. Black line: forward-afterward. Blue line: Right-Left. Yellow line: Up-Down

In Markov Random Field, the smoothness cost function enforces smooth labeling across neighboring hidden nodes. For each pair of neighboring cubes, we consider the compatibility of their label relationship and then define the smoothness cost energy of the lable assignment. In our research, the label relationship between neighboring cubes along the depth direction should also be considered. Although the 3D point cloud has been divided into $N{=}L{\times}H{\times}D$ cubes, the number of the cubes needed to be labeled is much smaller than N since some cubes are empty. Before any propagations, the empty cubes are discarded and the remains are used to form a MRF framework. However, some pairs of neighboring cubes in the MRF framework may have a long distance in the real word. Taking this factor into account, the smoothness term is defined by

$$E_{smoothness}(l_i, l_j) = \rho_d C(l_i, l_j) \tag{3.21}$$

where ρ_d is a binary function determines the dependence on the neighbor. If the distance between cube i and j is too far way, then $\rho_d = 0$. Otherwise, $\rho_d = 1$

Since in the 3D point cloud, the points only describe the surface of the objects, unlike 2D image, overlapping will not happen. Function $C(\cdot)$ determines the label compatibility of the neighboring cubes. $C(\cdot)$ is defined by the following rules: if the labels of a pair of cube (l_i, l_j) is compatible according to the model introduced next, then $C(l_i, l_j) = -1$. Otherwise, $C(l_i, l_j) = 1$. The consistent pair-wise model is shown in Fig.3.5, which is a 6-connected model. This model is developed from the one proposed for 2D image in [68], which is a 4-connected model indicating the up-down and left-right label relationship. In order to make it suitable for 3D scenes, an extra forward-afterward constraint was added . Apparently, for a same pair of labels, if their relative location

changes from left-right to up-down, the penalties obtained by $C(\cdot)$ will be different. This model is designed based on the fact that for the cube i labeled as O, cube j above it can be labeled as *Roof* or *Object*. Since there is no overlapping existed in a 3D scene, there is no reason to label it as *Wall*. However, if cube j is under cube i, then it can only be considered as *Object* or *Ground* since a *Ground* cube always has the lowest height. This model is used for examining the label compatibility and then calculating the smoothness cost energy.

3.5 Inference

Combining prior beliefs with observed evidence to form a prediction is called inference. Inference is used to solve the MAP estimation. We have divided our scene into several cubes, which are represented in terms of nodes of the MRF graphical model. For each node, datacost energy conditioned on the observation (feature) is calculated by (eq.3.19), and smoothness energy for a pair of neighboring nodes is computed by (eq.3.21). We aim to find the globally optimal assignment for the entire scene, that is finding $\mathbf{l} = [l_1, \cdots, l_N]^T$ minimizing the energy:

$$E(\mathbf{L}) = E_{datacost}(\mathbf{F} \mid \mathbf{l}) + E_{smoothness}(\mathbf{l}), \tag{3.22}$$

where $\mathbf{F} = [\mathbf{f}_1, ..., \mathbf{f}_N]$. According to this, we choose Loopy Belief Propagation (LBP) to solve the inference problem. LBP takes the form of a message-passing algorithm among nodes. In each iteration, the belief of a node is updated by adding the message passed from the neighboring nodes. Then after several iterations, for each node, the optimal label is the one with the strongest belief. Details of inference procedures by using LBP are given below:

step (1): For each node i, we use equation (3.19) to initialize the beliefs ${B_i}^0$ for each label, that is

$$B_i^0(l) = E_{datacost}(l), \tag{3.23}$$

where $l \in \{l_R, l_W, l_O, l_G\}$. Then we initialize the label for each cube by

$$l_i^0 = \arg\min_l(B_i^0(l)). \tag{3.24}$$

step (2): In iteration t, the belief $B_i^t(l)$ is updated by adding the messages passed from the neighboring nodes, that is

$$B_i^t(l) = B_i^{t-1}(l) + \sum_j M(i,j), \tag{3.25}$$

where $M(i,j)$ is the message passing from the neighboring node j to node i, that is

$$M(i,j) = B_j^{t-1}(l) + E_{smoothness}(l_i, l_j). \tag{3.26}$$

step (3): After the belief of all the nodes is updated, we update the label for each node according to equation (3.24).

step (4): Repeat steps (2) and (3) for K times, and the outputs $\mathbf{l}^K$ will be the final labeling result.

3.6 Experimental results

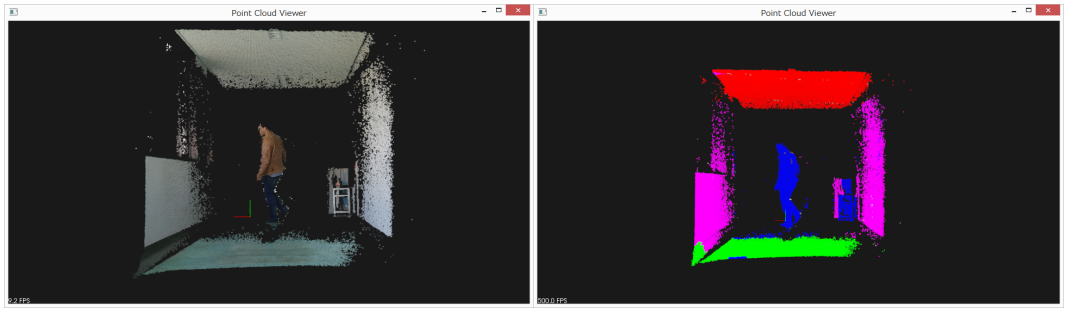
3.6.1 Dataset

Our experiment was implemented on three datasets taken from our laboratory. Each dataset consists of 80 frames of point cloud. The camera we used here is Kinect V2. However, since the camera exists some limitations, each frame contains a lot of noise points. Before any implementations, we first normalize both the width and height of the point cloud to be [-2.0m, 2.0m], then filter the points out of this range. Meanwhile, we took several single frames from different scenes for the purpose of comparing with other methods. All the datasets were taken from different data and under different illumination conditions. Details of the parameters used in this work are shown in table 3.1. η in eq.(3.13) was set to 75% empirically.

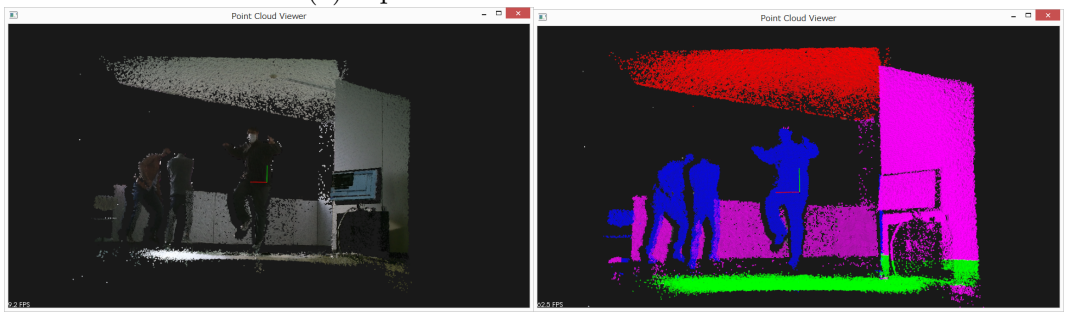
We first discuss the efficiency of the proposed method, then confirm the effectiveness through comparative experiments.

3.6.2 Efficiency

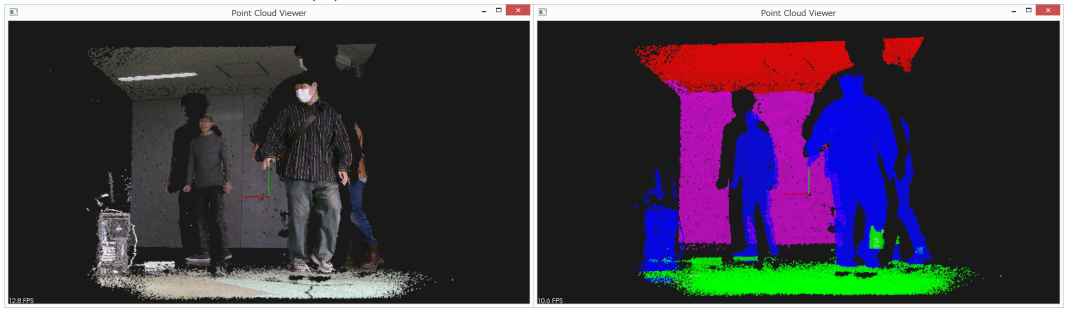
Figure 3.6 shows the labeling results by using the proposed method on the datasets, while table 3.2 shows the labeling accuracy rates. We first took less than 10 frames and hand-labeled for calculating $\tilde{\mathbf{r}}_X$, $\tilde{\mathbf{r}}_Y$, $\tilde{\mathbf{r}}_Z$ as well as matrix $\mathbf{\Sigma}_X$, $\mathbf{\Sigma}_Y$ and $\mathbf{\Sigma}_Z$. For increasing calculating speed and accuracy, the training set was online updated for every



(a) Input of dataset1 and the result



(b) Input of dataset2 and the result

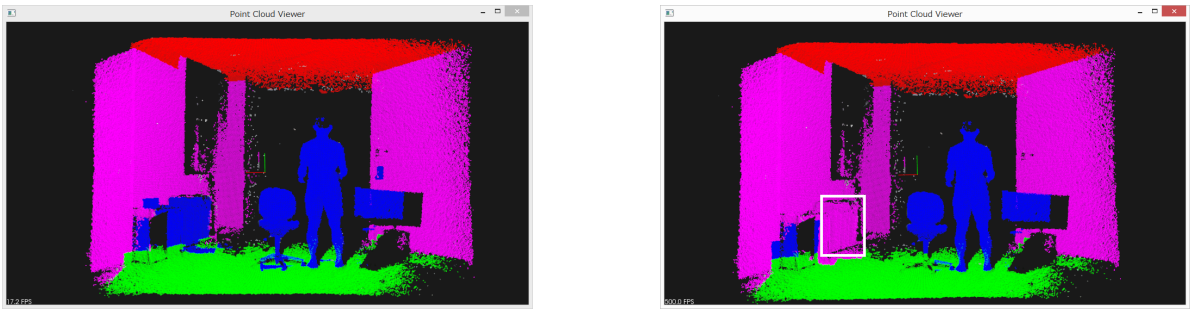


(c) Input of dataset3 and the result

Figure 3.6: Experimental results. Left part shows original frames taken from our dataset. Right part shows the labeling results obtained by using the approach proposed in this papaer

5 frames. On an average, the number of the cubes containing points was very low (12% of the whole number N), in which the number of the cubes labeled as *Object* was less than 20%. Accuracy of *Object* labeling is more important than the others since in many studies, it is always used as ROI thus potential objects such as pedestrian are supposed to be detected from it. From Fig.3.6 it can be observed that for the adjecent objects those have different categories (trash can and wall), it is more difficult to distinguish

them. However, even though the vertical distribution and depth distribution of such cubes might be similar, the horizontal distribution of them is different. For this reason, each bin boundary of $\mathbf{r}_X$, $\mathbf{r}_Y$, $\mathbf{r}_Z$ was designed to be uneven. Figure 3.7 shows a comparative result of using a changed bin boundary (bin 1 of $\mathbf{r}_x$ was changed to [0-0.3] to make the trash can contained within this region) and the trash can was mislabeled as wall, which was marked with a white frame. Increasing the dimension of SDF, that is, the number of histogram bins, could improve the accuracy for adjacent objects but bring numerous computation.



(a) boundary in Table 3.1 (b) changed boundary

Figure 3.7: Comparative results

Table 3.2: Labeling rates for each label

database 1	label	R	W	O	G
	number	130	266	74	96
	rate	99%	93%	93%	90%
database 2	label	R	W	O	G
	number	129	281	245	68
	rate	98%	95%	92%	85%
database 3	label	R	W	O	G
	number	115	206	225	75
	rate	95%	93%	90%	80%

The disadvantage of our method can be obviously observed that some cubes located at the edge of an object were more inclined to be labeled badly (e.g. the connected part between *Object* and *Ground*), which was caused by the size of the cube. If the scene was divided with a large size, points contained in the divided cube might belong to different

labels. As a result, all the points in such cubes will be labeled as the same category. Decreasing the size could lessen this problem, that is, enforcing the points contain in each cube to have the same label. However, it will make the extracted feature become weak since the number of points has been reduced and bring about more computation as well. For this reason, it is necessary to find an appropriate cube size for the balance. Numerous experiments have proved that setting the cube size as table 3.1 gives the best performance.

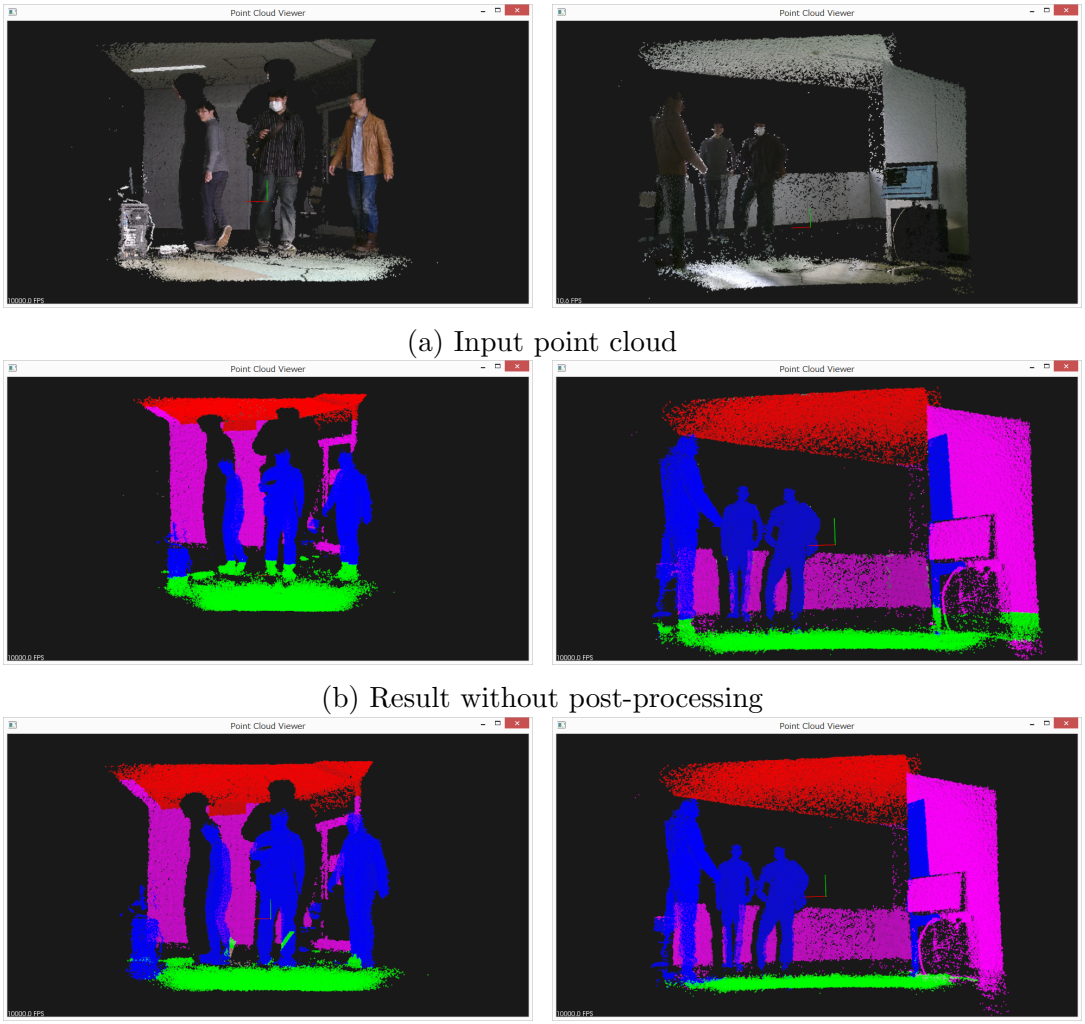
A post-processing was proposed to solve this problem essentially. After the labeling procedure, we gather all the cubes labeled as G and those adjacent to G to form a large set of points **P**. In this set **P**, we find the points belonging to G with the method proposed in Chapter 2, then for each remaining point p, if p belongs to $cube_i$, we check the label of the neighboring cube of $cube_i$ only along vertical direction, which is represented as $cube_{i(n)}$, and p will be assigned with the same label as $cube_{i(n)}$. It is reasonable that in the indoor scene, except for roof and ground, category of other objects along the vertical direction is continuous. Results of the post-processing are shown in Fig.3.8, where the performance has been improved to a certain degree.

Figure 3.9 and 3.10 show the labeling results of dataset 1 when the person is moving from a far and close distance respectively. We observed the performance when the person is moving from left to right. It can be seen that when the person is close to the trash can on the right part, the performance of labeling O is getting better. This is because when the person is far away from the trash can, the labeling of the trash can is only depended on the ground and wall, and the similarity of their features makes some cubes were mistakenly labeled as W. When the person is closing to it, the label revelance is enhanced so that makes the performance better.

We can also observe by comparing Fig.3.9 with Fig.3.10 that the performance is getting better when the person is moving close to the camera. It is caused by the property of Kinect 2.0. The 3D point cloud created by this camera has a characteristic that more points will be captured from a closer distance. According to this, features of the cubes closed to the camera are stronger, which led to a better result.

3.6.3 Comparison

In this subsection, we compared our approach with two schemes, (i), using VSH feature proposed in [68] with no context model, and (ii), using VSH feature and the proposed



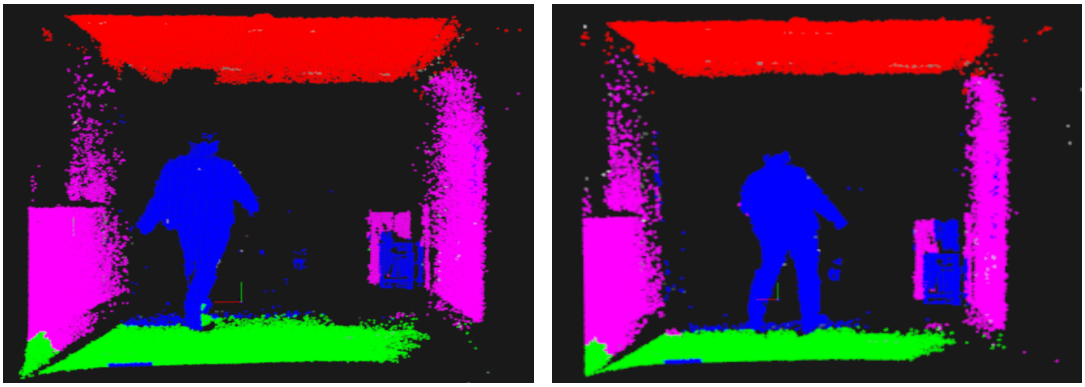
(a) Input point cloud

(b) Result without post-processing

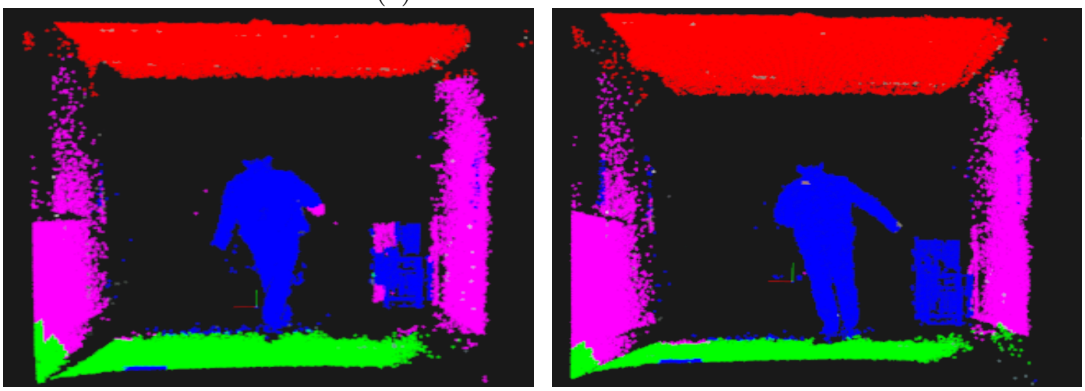
(c) Refined result

Figure 3.8: (a) shows the original point cloud. (b) shows the results obtained by using our method without any post-processing, where the connecting parts of person and ground were mistakenly labeled. By using the post-processing, the mistakenly labeled parts in (b) have been corrected as shown in (c)

context model in this paper. Results are shown in Fig.3.11 and the average labeling rates are shown in table 3.3. As illustrated, the primary factor caused scheme (i) to failure is that the context relationship was not considered, as well as the VSH feature is weak. Numbers of cubes were mislabeled as unappropriated categories. In scheme (ii), when using the appropriate context relationship, labeling rate increased a little, however, cubes holding similar VSH feature to each other were more inclined to be



(a) frame 37 and frame 41



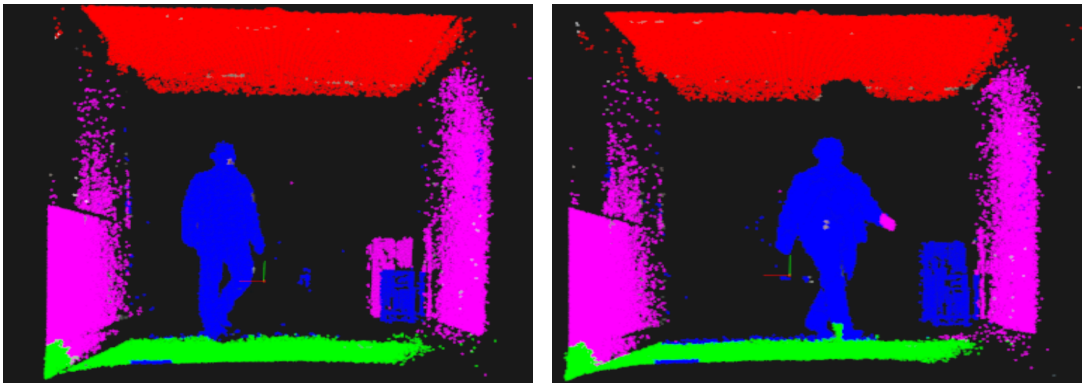
(b) frame 43 and frame 46

Figure 3.9: Labeling result when the person is moving from a far distance

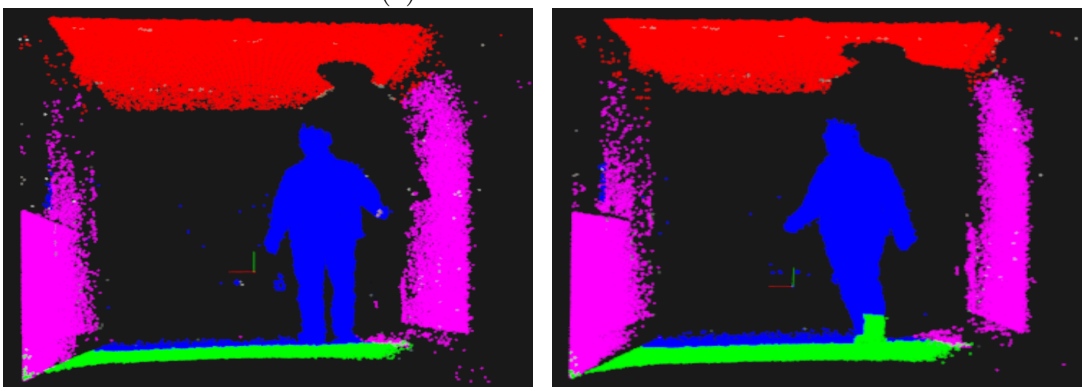
mistakenly labeled. For cubes (marked with frame in Fig.3.11 (c)) belonging to different categories, while holding similar distribution and average height, VSH feature turns out to be weak to distinguish their spatial characteristics, which resulted in mislabeled. In Fig.3.11 (d), as we discussed above, since the feature proposed in this chapter captured the whole spatial distribution, even for the cubes which have same VSH, it was easy to label by considering the other features.

3.7 Conlcusion and further research

In this chapter, we have presented an innovative approach for indoor scene labeling based on a Bayesian Framework. This approach incorporates several novel ideas: (i), we have proposed a method to learn a novel spatial feature vector called SDF that derives from the combination of three oriental distribution by using eigenvector decomposition



(a) frame 62 and frame68



(b)frame 75 and frame 78

Figure 3.10: Labeling result when the person is moving from a close distance

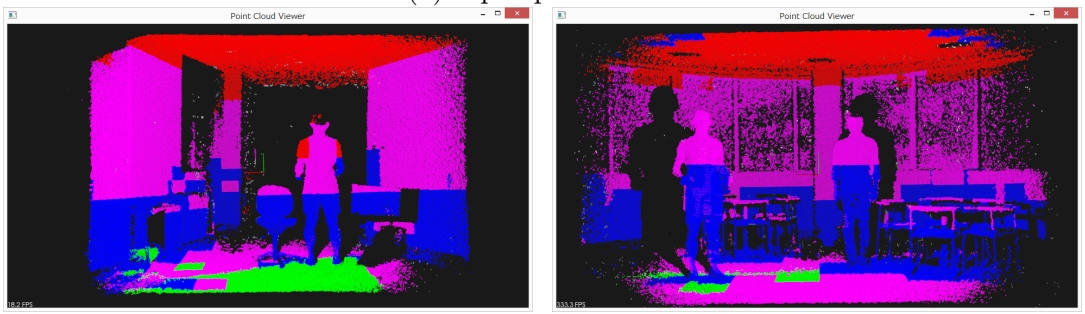
Table 3.3: Average labeling rate of comparative scene

VSH only	Label	R	W	O	G
	Rates	85%	62%	51%	32%
VSH+our model	Label	R	W	O	G
	Rates	88%	68%	62%	29%
SDF+our model	Label	R	W	O	G
	Rates	98%	95%	90%	93%

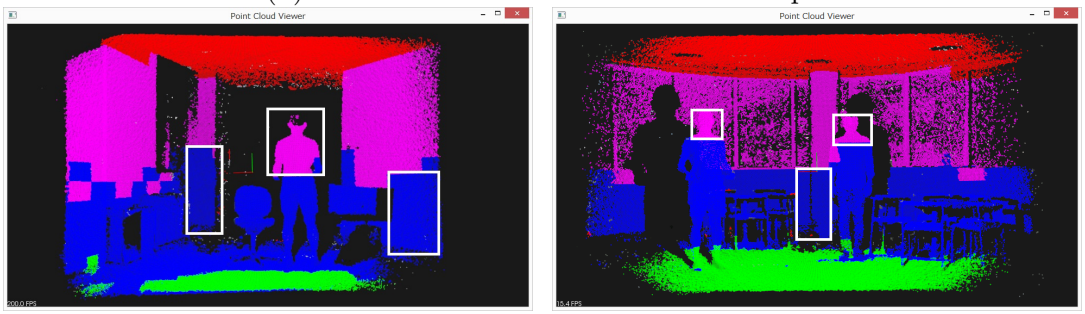
and sub-space combination. (ii), we have designed a 6-connected model to represent the label relationship in 3D scene, then developed a corresponding smooth cost function that is used for Markov Random Field. Extensive experiments have been performed and the results confirmed the performance and the effectiveness of our approach.

As we discussed in section 3.6, a disadvantage of this approach is that the size of cube has an influence on the final results. We have proposed an improved approach additionally to solve this problem. Another disadvantage is that the boundary of each histogram bin was designed manually and needed to be changed according to different scene. In our future work, we will focus on resolving these problems. Meanwhile, since we have been able to label *Object*, based on which we can develop our approach for more challenging research such as people detection and moving objects tracking.

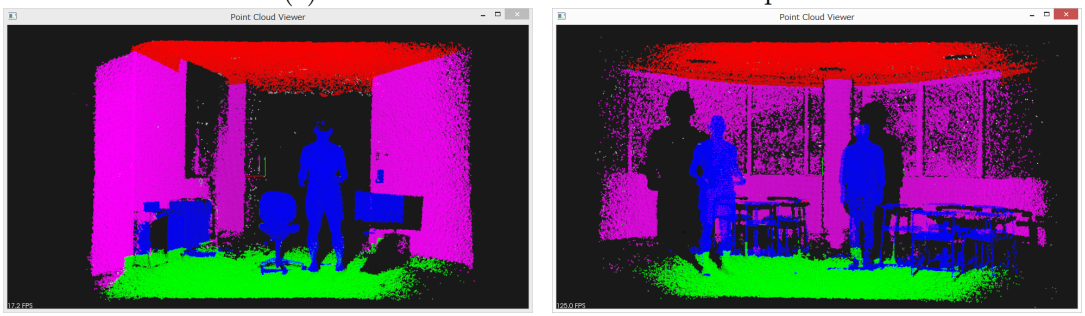
(a) Input point cloud



(b) VSH feature with no label relationships



(c) VSH feature with label relationships



(d) SDF feature with label relationships

Figure 3.11: Comparisons with other methods. First row shows the original point cloud. Second row shows the results obtained by using VSH feature with no label relationships. Third row shows the results obtained by using VSH feature and the context model proposed in this chapter. The bottom row shows the results obtained by using SDF feature and the context model designed in this paper.

4

3D Indoor Scene Labeling with Conditional Random Field

In this chapter, a Bayesian framework is proposed to label the objects in the 3D scene. In order to make the labeling performance robust against camera rotation, we develop a set of rotation invariance feature vectors, which are discriminative enough to distinguish different objects as well. Moreover, the spatial relevance between different categories should be captured more precisely. In the previous chapter we have proposed such a model describing the label relationships. However, according to its own limitations, this model is not applicable for all the 3D scenes. Alternatively, we aim to learn a more effective model from the training set.

Same to Chapter 3, the labeling task is solved by using the graphical model. However, in this time we aim to use more than one feature in purpose of improving the performance. According to this consideration, Conditional Random Fields (CRF) is regarded to be more suitable because of its ability of using several features. Moreover, CRF is more capable to smooth the independency results.

In order to achieve a robust labeling performance, the following key ideas are used in this thesis:

1. Using normal-deviation feature to develop a high dimensional feature vector which is robust against camera rotation.

2. Using an supervoxel over-segmentation task to divide the 3D scene, ensuring that rich features can be exploited from each voxel.

3. Learning the label relevance from the training set.

4. Using normal-deviation feature to define the pairwise potential of CRF.

5. Using normal-deviation feature for people detection.

Each key idea will be discussed in the following parts of this chapter. The content of this chapter is organized as follows:

1. In section 4.1, we give an overview of our 3D indoor scene labeling algorithm.

2. In section 4.2, we give the details of each procedure in our algorithm, including the feature we proposed, definition of unary potential and pairwise potential of CRF, and the method of learning label relevance.

3. In section 4.3, a person detection task using the result of the labeling is proposed.

4. In section 4.4, we present the experimental results comparing with other approaches .

5. In section 4.5, we make a short conclusion and discussion.

4.1 Overview

The overview of our proposed framework is shown in Fig.4.1, which consists of two layers: a scene labeling layer followed by a moving person detection layer. In the first layer, a 3D indoor scene in the form of point cloud is generated by using a RGB-D camera and rotated by a random angle. Then we use an advanced over-segmentation approach to cluster the point cloud into to a number of supervoxels with similar color, appearance and surface direction. From each supervoxel, we extract the proposed feature vector which captures the appearance, color, related position, and related direction properties. After that, the feature is passed into a pre-trained classifier to calculate the posterior probability of each label, which is used as the initialized unary potential of the Conditional Random Fields model. A new pairwise model is defined to improve the performance of labeling, for which the parameters and the context relation are learned from the training set. The CRF labeling is solved by using an efficient algorithm called Mean Field Approximation.

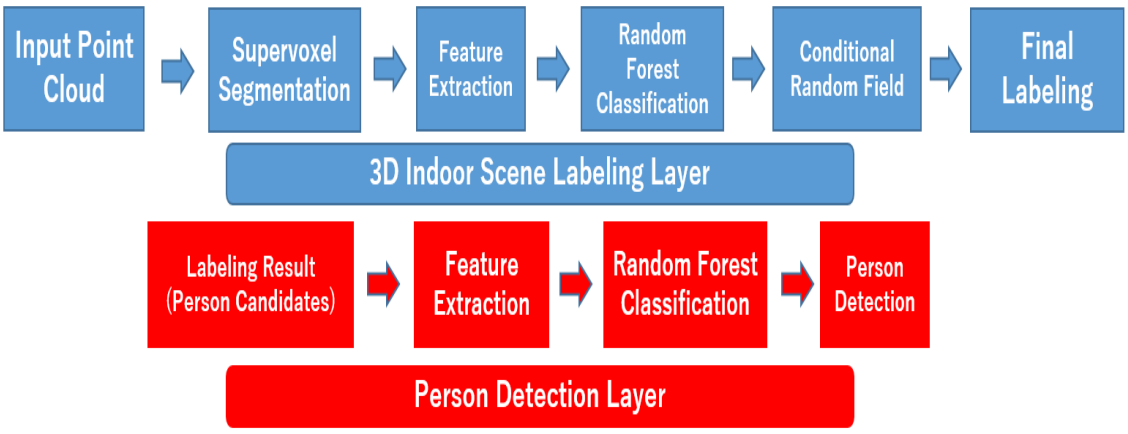


Figure 4.1: Overview of our 3D indoor scene and Person Detection framework. Details will be give in the next section.

As the outputs of the first layer, objects labeled as *Candidates* are passed into the second layer for person detection. A binary classifier is called to classify the *Candidates* as Person and Non-Person with another proposed feature vector which is discriminative to rigid and non-rigid objects. All the details of each procedure will be discussed in the following sections.

4.2 3D indoor labeling

Our scene labeling layer is handled by a CRF framework. A general CRF model is characterized by a Gibbs distribution:

$$P(\mathbf{y}|\mathbf{V},\boldsymbol{\theta}) = \frac{1}{Z(\mathbf{V},\boldsymbol{\theta})} \exp\left(-\sum_i \Psi_i(y_i|\mathbf{v}_i,\boldsymbol{\theta}) - \sum_{i\neq j} \Psi_{ij}(y_i,y_j|\mathbf{v}_i,\mathbf{v}_j,\boldsymbol{\theta})\right) \tag{4.1}$$

where $(\mathbf{V},\boldsymbol{\theta})$ is the observations (features) and $\mathbf{y}$ is the random variables (label) of CRF. $Z(\mathbf{V},\boldsymbol{\theta})$ is the normalization function. $\boldsymbol{\theta}$ is the set of parameters. The unary potential Ψ_i can be calculated independently for each supervoxel by a classifier, which will be discussed later. Ψ_{ij} is the pairwise potential we re-designed in this paper. This formula can be expressed as the Gibbs energy:

$$E(\mathbf{y}|\mathbf{V},\boldsymbol{\theta}) = \sum_i \Psi_i(y_i|\mathbf{v}_i,\boldsymbol{\theta}) + \sum_{i\neq j} \Psi_{ij}(y_i,y_j|\mathbf{v}_i,\mathbf{v}_j,\boldsymbol{\theta}) \tag{4.2}$$

and the optimal label assignment $\mathbf{y}^*$ can be solved by minimizing the Gibbs energy as:

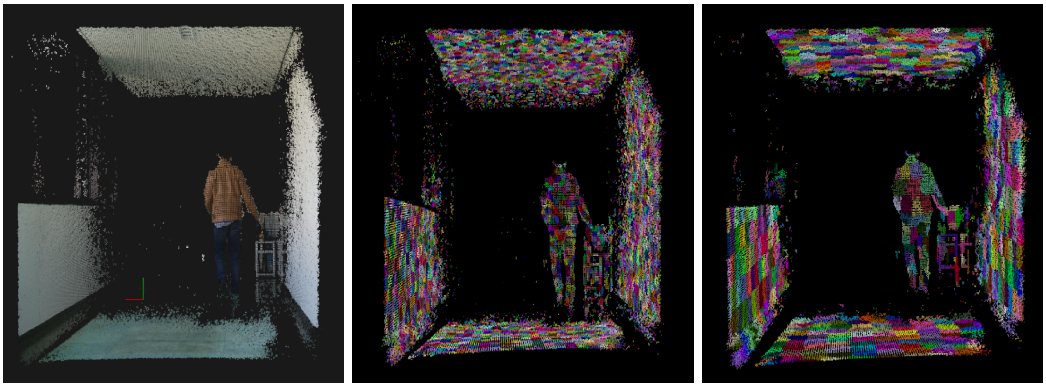
$$\mathbf{y}^* = \arg\min_{\mathbf{y}} E(\mathbf{y}|\mathbf{V}, \boldsymbol{\theta}) \tag{4.3}$$

4.2.1 Supervoxel clustering

In order to train an effective classifier, the features used for training are extracted from a set of points having the same properties (e.g. color, orientation). The simplest way to create such sets is to divide the whole point cloud into several cubes manually. However, if the cube size is too large, it would enforce the points in the cube having different categories, which results in a weak feature extraction. Although reducing the size could prevent that situation from happening to a certain degree, it would bring about more computations. Moreover, the number of the points contained in each cube will be reduced, which makes the feature weaker. To solve this problem, we use an effective over-segmentation method proposed in [80] to divide the point cloud into several supervoxels (one of the over-segmentation result is shown in Fig.4.2). This method uses a region growing based method to cluster the point cloud under a voxel octree structure. The advantage of using supervoxel is that they are evenly distributed across the 3D space, and they cannot cross boundaries unless the underlying voxels are spatial connected. More importantly, points in each supervoxel have the same properties. The performance of this method depends on the voxel resolution r_v, the seed resolution r_s and three weighting parameters w_c, w_n, w_s, which control the influence of color, surface normals and spatial distances for each cluster. Since our work is based on a voxel-represented cloud, accuracy of the clustering will affect the final result.

4.2.2 Feature extraction

After the whole 3D point cloud is clustered into supervoxels, for each of them, we extract a feature vector capturing the properties of appearance, color, and related position. We aim to use a set of features which are invariant to camera rotation to ensure the feature vector robust. A 33-dimention FPFH [61] is used to capture the appearance of the voxel since we have discussed its invariance property above. Apparently, considering the rotation invariance, spatial feature developed from the coordinate information becomes unstable. However, the relative position in the whole scene is unchanged even after



(a) input point cloud (b) by using a small r_s (c) by using a large r_s

Figure 4.2: Supervoxels oversegmentation results with different parameters. (b) and (c) show the results brought by using r_s=9cm and r_s=18cm on the input point cloud (a).

Table 4.1: Details of the feature vector used for calculating unary potential

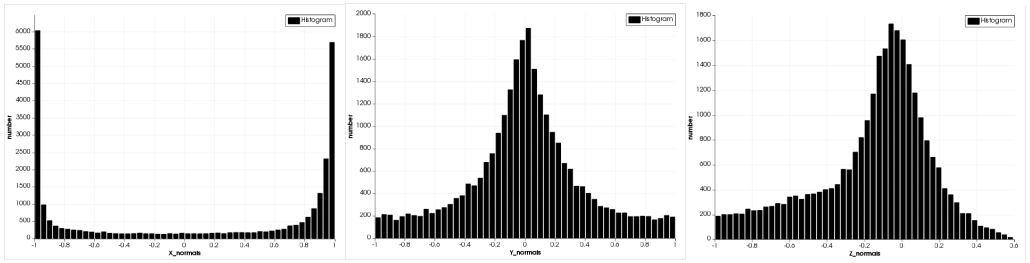
feature	dim
Normal-deviation	3
Color in CIELAB space	3
Distance from voxel centroid to cloud centroid	1
FPFH in [61]	33
Total number of features	40

rotation. For each voxel, the distance from its centroid point to that of the whole scene are calculated and used as a feature.

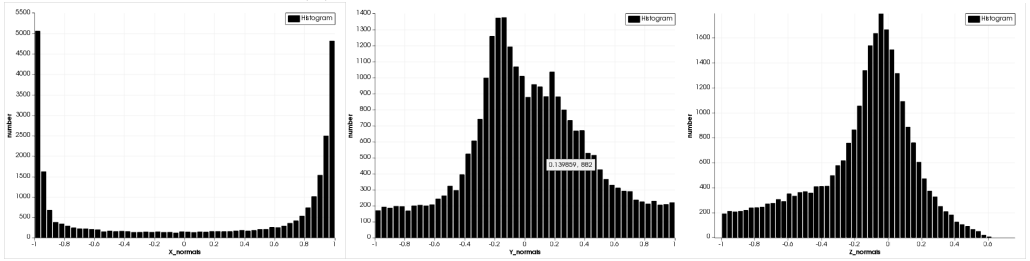
Another useful invariant feature is developed from the distribution of surface normal vectors. Given a geometric surface, it is usually trivial to infer the direction of the normal as the vector perpendicular to the surface. For the same object, after being rotated, the normal of each point varies with the same rotation angle. Figure 4.3 shows the distribution of surface normal calculated for each point belonging to two types of objects before and after rotation. Although the mean and range of the distribution has been changed, the scale remains similar. We use the deviation of such distribution to emphasize the rotation invariance and name this feature as normal deviation in this work.

Details of components of our developed feature vector is shown in Table 4.1. This

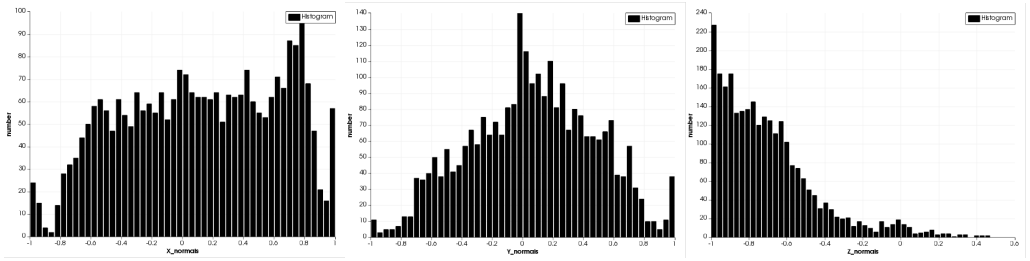
feature vector is used for training the classifier and calculating the unary potential that will be introduced in the next subsection.



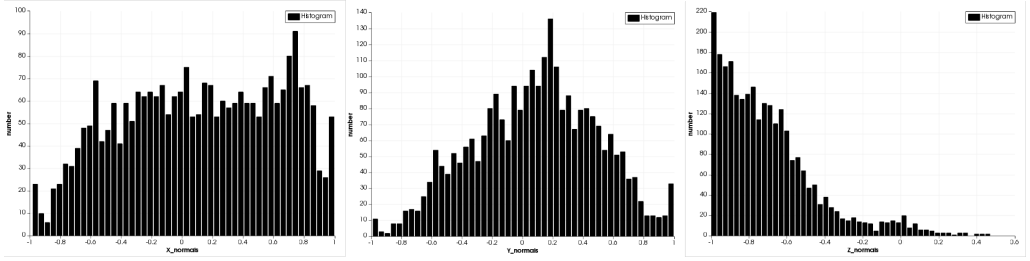
(1) regular objects before rotation



(2) regular objects after rotation



(3) irregular objects before rotation



(4) irregular objects after rotation

Figure 4.3: Histograms of the normal vectors along x, y and z direction (from left to right) of regular and irregular objects before and after rotation. For each histogram, the vertical axis denotes the value of the bin(the number of points) and the horizontal axis shows the coordinate information

4.2.3 Unary potential

For each voxel, we use an efficient classifier to calculate the posterior probability $p(y|\mathbf{x})$ of each label $y \in \{l_1, l_2, \cdots, l_M\}$ conditioned on the extracted feature vector $\mathbf{x}$, then we use this probability to initialize the unary potential of CRF defined in eq.(4.1). In our research, the classifier is chosen as Random Forest (RF) which has some significant properties: 1) it can handle very high dimensional feature vectors as well as large number of training examples. 2) for each input vector, RF gives the output as a probabilistic label distribution, from which $p(y|\mathbf{x})$ can be calculated directly. 3) RF has a remarkable computing speed while maintaining high accuracy, which is an important factor in determining the final result.

The RF classifier is pre-trained before any procedures from the datasets. Daniel Wolf in [69] trained the classifier from the datasets enlarged by adding 10 rotated over-segmentions per input point cloud to make the classifier invariant to camera rotation. In our research, in order to prove the robustness of the feature vector we have developed, each point cloud used in the training set is obtained by a fixed camera.

Details of classification for each voxel with the extracted feature vector are shown as follows: first, each decision tree outputs a value 1 (vote) for the decided label and 0 for the others. Then the number of the votes for each label l as n_l is calculated over all the trees. Finally, probability of each label l is represented as the proportion of the votes n_l to the number of the trees N_T:

$$p(y = l|\mathbf{x}) = \frac{n_l}{N_T} \tag{4.4}$$

The unary potential is initialized as:

$$\Psi_i(y_i) = -\log\{p(y_i = l|\mathbf{x})\} \tag{4.5}$$

Since this posterior probability is calculated independently, that is, without considering the label relationship among neighboring voxels, the labeling result solved by MAP estimation is generally noisy and inconsistent. In CRF, the pairwise term takes these relationship into account and is able to refine the performance. Modeling the pairwise will be introduced in the next subsection.

4.2.4 Pairwise potential

Pairwise potential denotes the cost of assigning each pair of labels to the neighboring voxels, which makes it enable to smooth the noise caused by independency. When the features are not discriminative enough to predict labels correctly, taking account of the context relationships between different labels can improve the labeling performance significantly because it eliminates ambiguity. In this paper, we use a linear combination of m kernel functions to model the pairwise potential, which is given by:

$$\Psi_{ij}(y_i, y_j|\mathbf{v}_i, \mathbf{v}_j, \theta) = \sum_m \mu^m(y_i, y_j|\theta) w^{(m)} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j), \tag{4.6}$$

where $w^{(m)}$ is the linear combination weights of each kernel. $\mu^m(\cdot)$ is a function that examines the compatibility of a pair of labels. Each kernel $k^{(m)}(\mathbf{f}_i, \mathbf{f}_j)$ is define as a Gaussian kernel:

$$k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) = \exp(-\frac{1}{2}(\mathbf{f}_i - \mathbf{f}_j)^T \mathbf{\Sigma}^{(m)}(\mathbf{f}_i - \mathbf{f}_j)), \tag{4.7}$$

where $\mathbf{f}_i$ and $\mathbf{f}_j$ are feature vectors for voxel $\mathbf{v}_i$ and $\mathbf{v}_j$ in an arbitrary feature space. Note that this feature vector is not the same as the one introduced in subsection Unary Potential. $\mathbf{\Sigma}^{(m)}$ is a symmetric, positive-definite precision matrix, which defines the shape of each kernel $k^{(m)}$.

Hermans in [73] has proposed a two-kernel potential consists of appearance potential and smoothness potential. The appearance potential is given by:

$$k^{(1)} = \exp(-\frac{|\mathbf{p}_i - \mathbf{p}_j|}{2\theta_\alpha^2} - \frac{|\mathbf{c}_i - \mathbf{c}_j|}{2\theta_\beta^2}), \tag{4.8}$$

where $\mathbf{p}$ is the 3D position and $\mathbf{c}$ is the color in CIELAB space of each voxel. This model is used to specify the connections between voxels having similar appearance and color along a wide range. θ_α and θ_β are the parameters specifying the range in which voxels with similar coordinates and colors will affect each other, respectively.

The second potential called smoothness potential is given by:

$$k^{(2)} = \exp(-\frac{|\mathbf{p}_i - \mathbf{p}_j|}{2\theta_\gamma^2} - \frac{|\mathbf{n}_i - \mathbf{n}_j|}{2\theta_\delta^2}), \tag{4.9}$$

where $\mathbf{n}$ is the surface normal vector of each voxel. This potential operates in a small range to examine the compatibility between labels according to that close voxels with

similar surface orientations are more likely to have the same label. θ_γ specifies the range which is much smaller than θ_α and θ_δ defines the degrees of similarity of the normals.

An obvious disadvantage of the smoothness potential is that it only applies to objects with regular shapes. However, in this work, we pay more attention to improve the labeling rate of person, the shape of which is irregular. For a pair of neighboring voxels of the object "person", even though the normals of their surfaces are quite different, they still belong to the same label. Since the surface of each voxel can be thought as planar, the normal vector of each point in it has the same direction. For this reason, we give another potential called second-smoothness potential which is given by:

$$k^{(3)} = \exp(-\frac{|\mathbf{p}_i - \mathbf{p}_j|}{2\theta_\gamma^2} - \frac{|\mathbf{d}_i - \mathbf{d}_j|}{2\theta_\epsilon^2}), \tag{4.10}$$

where $\mathbf{d}$ is the normal-deviation of each voxel. θ_ϵ defines the degree of similarity between two normal-deviations, and it is smaller than θ_δ. It is used to reduce the classification noise for irregular objects, which can not be solved by $k^{(2)}$.

Label compatibility functions $\mu^{(m)}$ are defined separately. We use a simple Potts model to define $\mu^{(2)}$ and $\mu^{(3)}$ for the two smoothness potentials as:

$$\mu^{(2)}(y_i, y_j) = \mu^{(3)}(y_i, y_j) = \begin{cases} 1 & (y_i \neq y_j) \\ 0 & (otherwise) \end{cases} \tag{4.11}$$

For the appearance potential, $\mu^{(1)}$ should capture context relations between different labels across larger range. In our paper, $\mu^{(1)}$ is learned from the training set and should be invariant to camera rotation. Method for learning this model is given as follows: first, in each point cloud of the training set, for every voxel labeled as l_i, we define a sphere centered on this voxel with the radius $r = \theta_\alpha$. Then we calculate the number of the voxels in this sphere labeled as $l_i = \{l_1, \ldots, l_M\}$ respectively, and generate a global histogram $\mathbf{h}_{l_i} = [N_{l_1}, N_{l_2}, \ldots, N_{l_M}]^T$ over all the voxels, where N_{l_i} is the number of the voxels belonging to label l_i. Each histogram is normalized and then used to construct a $M \times M$ symmetric matrix $\mathbf{H} = [\mathbf{h}_{l_1}, \ldots, \mathbf{h}_{l_M}]$. The finally learned matrix is averaged from the whole training set. Element in the matrix shows the likelihood of assigning a pair of labels to two different voxels within the randge $r = \theta_\alpha$. This matrix will not be affected by camera rotation.

Since we use a linear combination of Gaussian kernels to model the pairwise potential, Mean Filed Approximation (MFA), which is an efficient inference method proposed in [81], is employed to solve the labeling. We approximate $P(\mathbf{y}|\mathbf{X})$ by a distribution $Q(\mathbf{y})$ that minimizes the KullbackLeibler divergence $D(Q||P)$[82], such that Q is a product over its marginals $Q(\mathbf{y}) = \prod_i Q_i(y_i)$. This distribution is approximated by using an iterative approach with linear complexity in the number of voxels. After a number of iterations, optimal labeling for each voxel can be obtained by setting the label $l_i = \arg\max_l Q_i(l)$. This inference method has a linear complexity in the number of voxels and converges very fast such that it can deal with a large number of variables in a short time. After the inference, our scene is labeled as four categories: $Roof$, $Wall$, $PersonCandidates$ and $Ground$ (R, W, C, G).

4.3 Person detection

Voxels belonging to $Person$ are detected from the objects labeled as $PersonCandidate$ by a classifier. First we use a simple filter to remove the voxels mislabeled as C, which can be treated as noise. The rest voxels consist of two categories: $Person$, the shape of which is irregular, and non-person, the shape of which remains still. For this consideration, we use an appearance feature vector for binary-classification. Based on FPFH, a more efficient feature vector named Viewpoint Feature Histogram (VFH) has been developed by in [65]. It added viewpoint variance to FPFH by using the viewpoint vector direction. It is also a global descriptor that captures the shape information greatly with low computational cost. Moreover, this feature is invariant to scale and rotation.

Similar to before, we combine color and normal-deviation with VFH to make the feature vector discriminative and more robust to camera rotation. Details of the feature vector are shown in Table 3. Since it is a high dimensional feature, we choose Random Forest Classifier according to its properties we discussed above. The classifier is trained from the training set created by a fixed camera (no rotations).

Table 4.2: Details of the feature vector used for person detection

feature	dim
Normal-deviation	3
Color in CIELAB space	3
VFH in [65]	308
Total number of features	314

4.4 Experimental results

4.4.1 Dataset and parameters

Dataset

Our experiment was implemented on three datasets taken from our laboratory. Each dataset consists of 80 point clouds containing moving person. All the datasets were created by a fixed RGB-D camera without any moving or rotations. The camera we used here is Kinect v2. We used the first 30 frames of point cloud without any rotations to train the contextual relations introduced above as well as the RF classifiers used for unary potential and person detection, respectively. For evaluation, each point cloud of the remaining frames was rotated with a random angle between $[-20°, 20°]$. To improve the efficiency of person detection, objects adjacent to wall are considered as label "Wall" such as bookshelf, board and clock. All the Datasets were taken from different dates under different illumination conditions.

Parameters setting

For the over-segmentation, we set the voxel resolution r_v to 1.8 cm and the seed resolution r_s to 9 cm. These settings were decided empirically considering the trade-off between speed and accuracy, so as to make each segmented voxel contain enough information. For the RF classifiers used in unary potential and person detection, the maximum number of trees for training N_t was set to 100 and 50 respectively. The maximum tree depth m_d was set to 30. Minimum samples required at a leaf node for it to be split m_s was set to 20. For CRF, the weights $w^{(m)}$ can be learned through the inference when using MFA. In the appearance potential, θ_α was set to 1 m. For

Table 4.3: Parameters used in our research (The definition of all the parameters were mentioned in the above sections)

Over-segmentation	r_v	r_s	w_c	w_n	w_s
	1.8cm	9cm	0.4	1	0.2
RF for unary	N_t		m_d		m_s
	100		30		20
RF for Detection	N_t		m_d		m_s
	50		30		20
Pairwise Potential	θ_α	θ_β	θ_γ	θ_δ	θ_ϵ
	1m	(12,3,3)	15cm	0.05rad	0.1rad

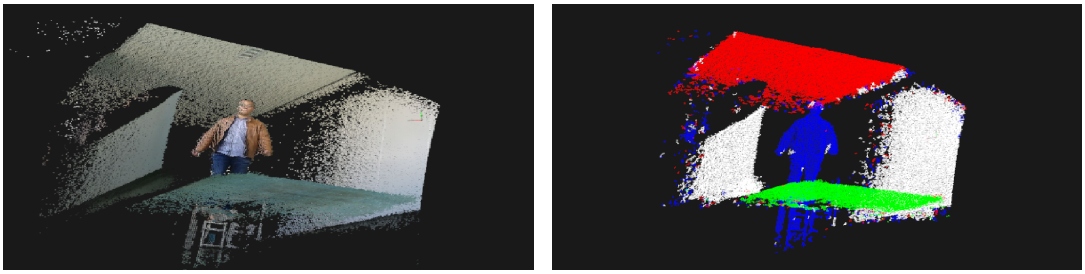
the color similarity parameters, $\theta_{\beta,L}$ was set to 12 for the L channel, and $\theta_{\beta,ab}$ was set to 3 for the a and b channels. In the smoothness, θ_γ was set to 15cm and the normal similarity θ_δ was set to 0.05 rad, while θ_ϵ was set to 0.1 rad. Iteration for inference was 5. Setting for all the parameters used in our research are given in Table 4.3, which is decided empirically based on the experimental results.

4.4.2 Evaluation and comparison

Performance

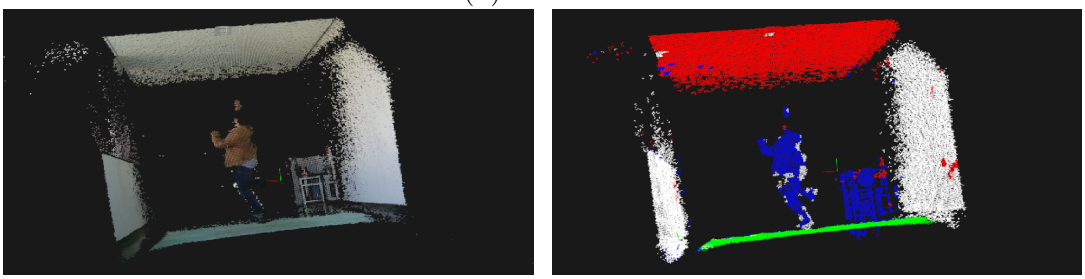
We first show the results obtained by using our algorithm on the three datasets. we used Red, White, Blue and Green to show the label of R, W, C and G. Figure 4.4 and Fig.4.5 show the results of dataset 1. Each frame was rotated by a random angle before labeling. The result wasn't affected by the rotation. However, since we used supervoxel to divide the scene, the size of each voxel is unequal. For some small voxels containing less than 10 points, the features inside them is very weak, which makes them labeled ambiguously.

Figure 4.6 and Fig.4.7 shows the results of dataset 2. This dataset was constituted by a simple indoor background and several moving people. The C objects were labeled with high accuracy, while others were more inclined to be mis-labeled. It was caused by that when people changed their position, the label revelance between them and other objects was varied, which affected the results.

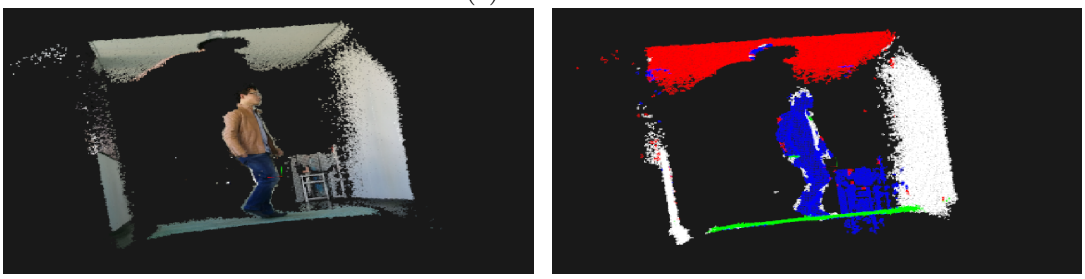


(a) frame 35

(b) frame 40



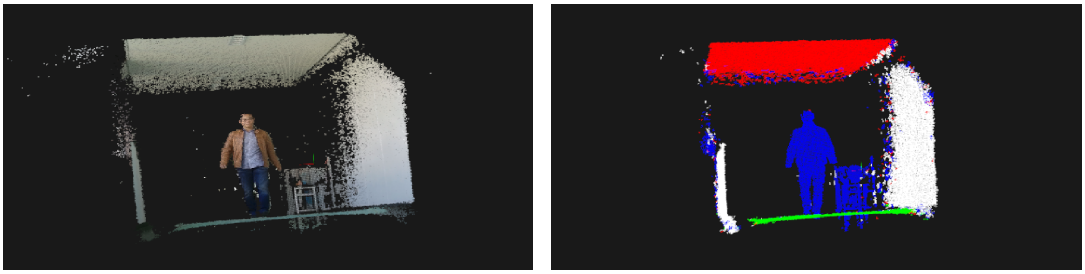
(c) frame 48



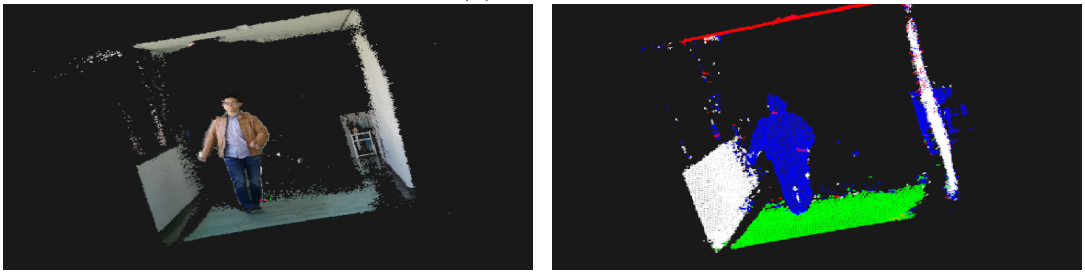
(d) frame 52

Figure 4.4: Labeling results of dataset1 (Part 1)

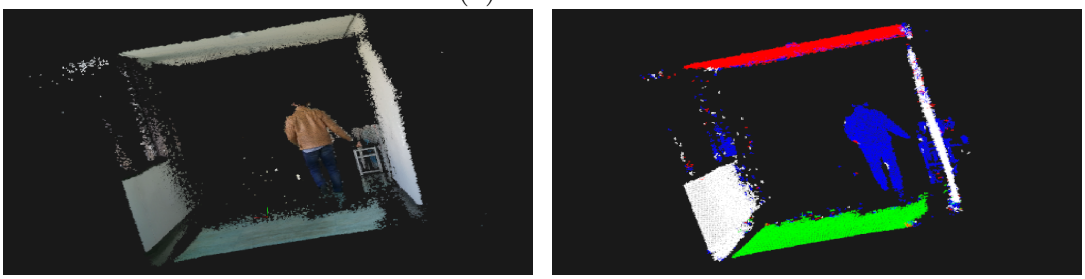
Figure 4.8 and Fig.4.9 shows the results on dataset 3, which constists of people moving from far to close in a complex background. The results of this dataset reached the best performance. However, in Fig.4.8 (a), when the person was standing closed to the wall, some parts were mistakenly labeled. It was caused by the similarity of their



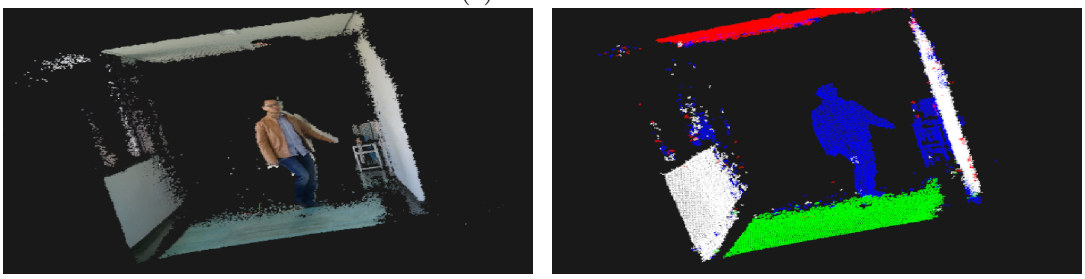
(a) frame 57



(b) frame 61



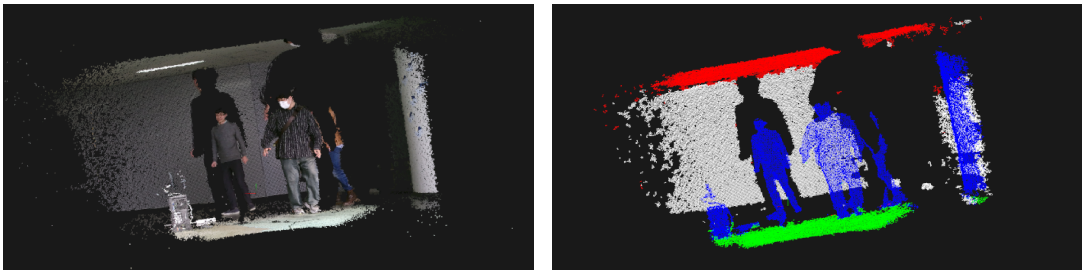
(c) frame 68



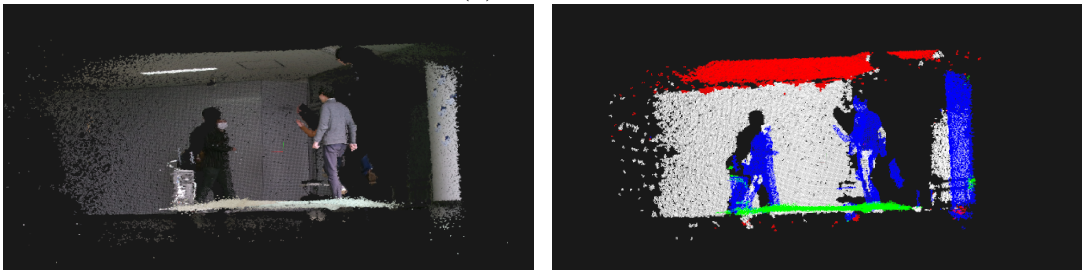
(d) frame 75

Figure 4.5: Labeling result of dataset1 (Part 2)

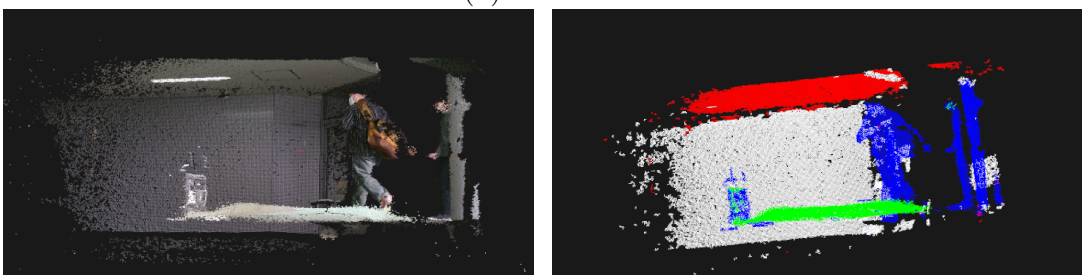
features. In Fig.4.8 (b), the same person was correctly labeled when leaving the wall.



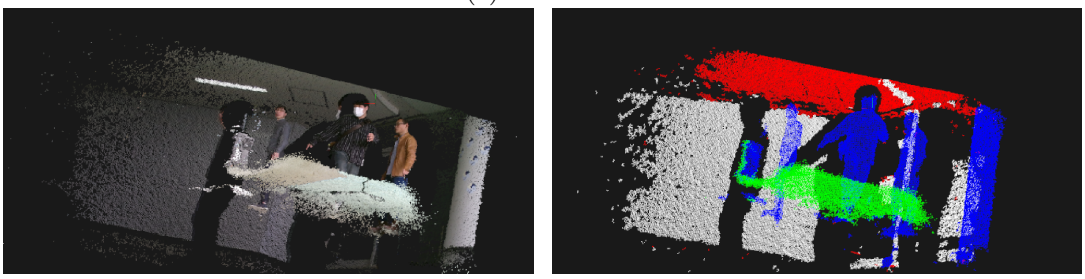
(a) frame 31



(b) frame 37



(c) frame 42

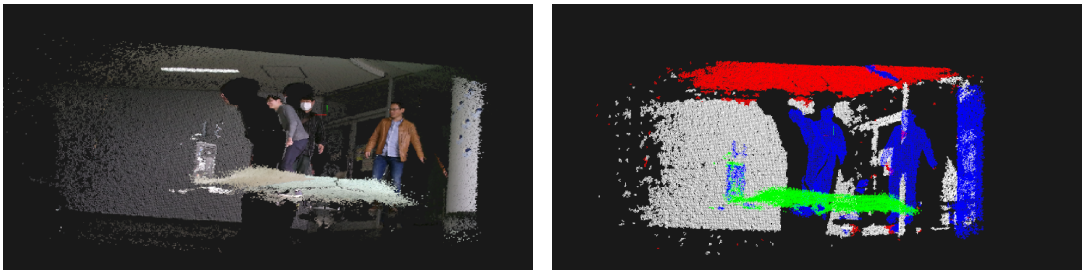


(d) frame 47

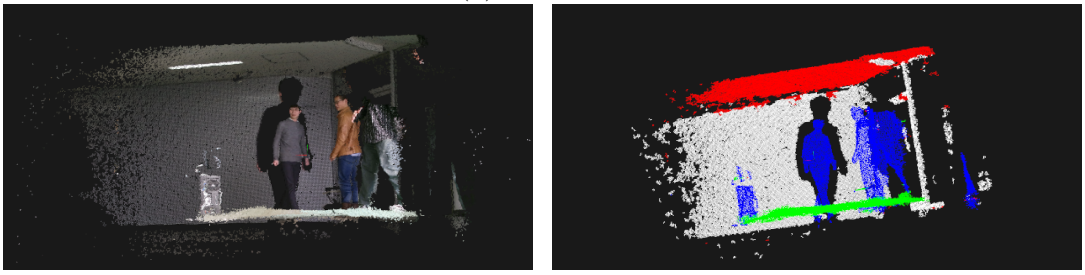
Figure 4.6: Labeling results of dataset2 (Part 1)

Comparisons

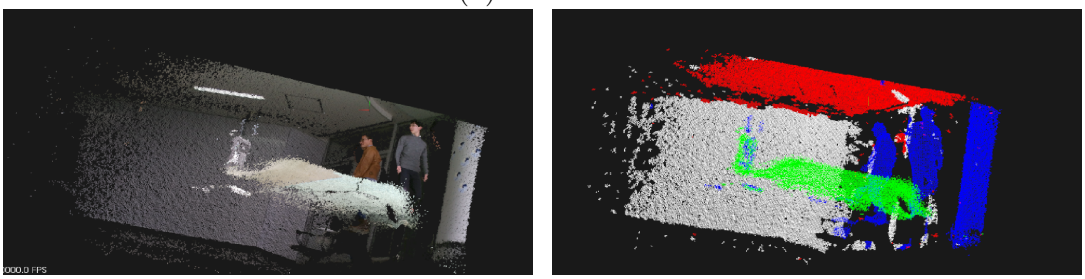
To evaluate our framework for labeling, first, we introduce another feature vector consists of position, color and FPFH. In our research this feature vector has a great effi-



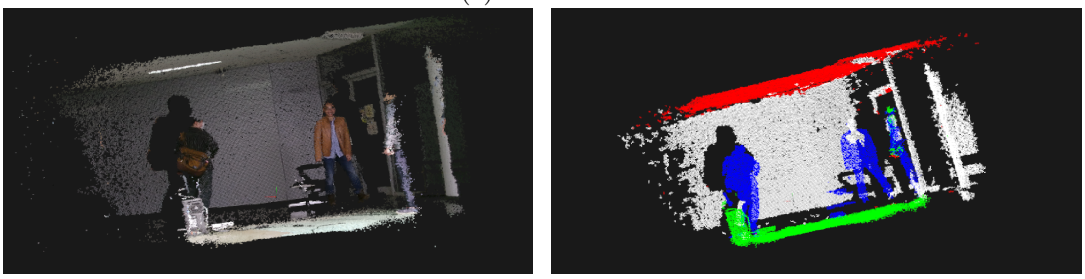
(a) frame 55



(b) frame 61



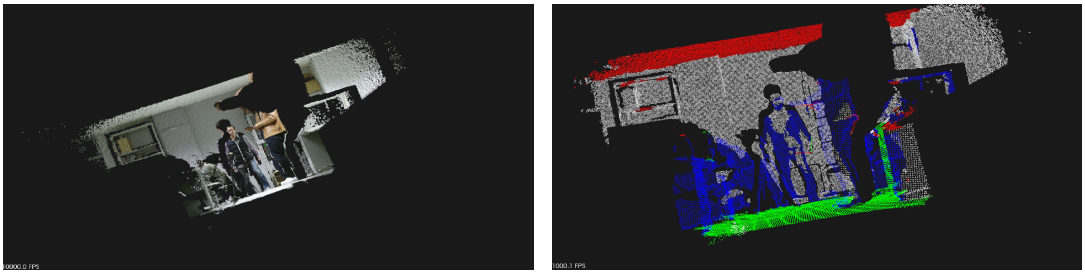
(c) frame 67



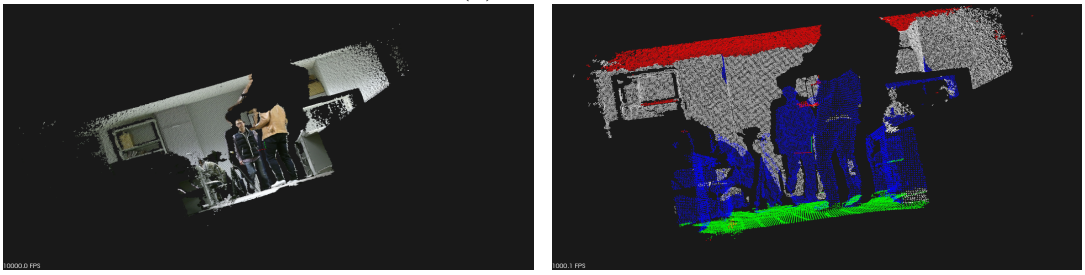
(d) frame 75

Figure 4.7: Labeling result of dataset2 (Part 2)

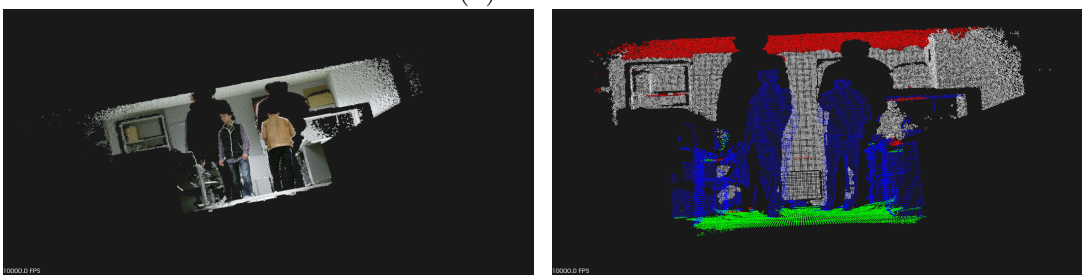
ciency that using RF without any refinement can reach a high accuracy. However, it lacks of robustness against camera rotation. This feature vector was used for comparison to evaluate the effectiveness of the proposed feature. We used two configurations of experiments: first, we compared labeling results by using the two feature vectors



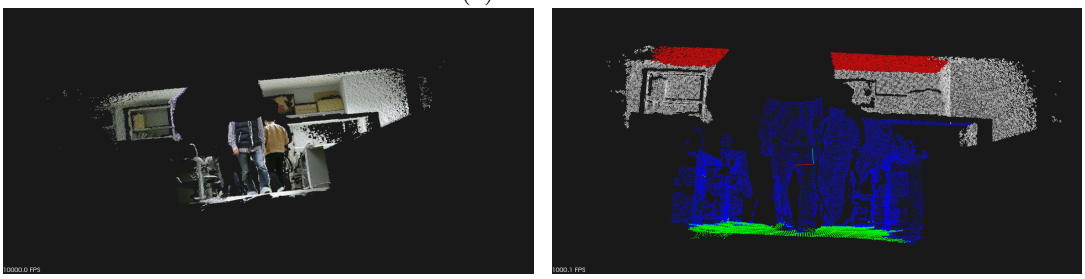
(a) frame 30



(b) frame 33



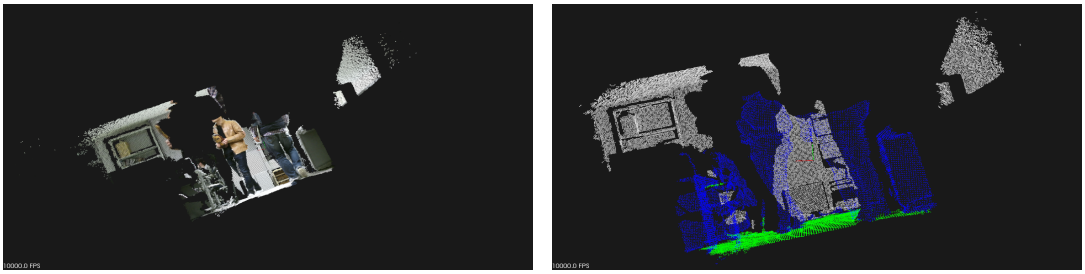
(c) frame 38



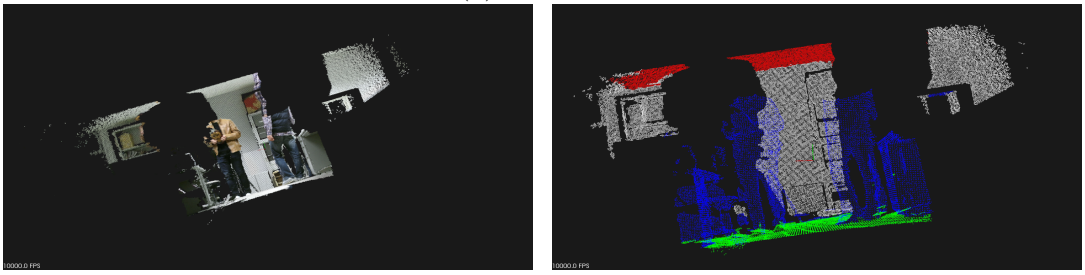
(d) frame 45

Figure 4.8: Labeling result of dataset3 (Part 1)

on a dataset with no rotation. Then we compared the labeling results on the same dataset after being rotated by an arbitrary angle. Labeling results with and without rotations are shown in Table 4.4. In the case of no rotations, position vector has a better quality than ours since the position features capture the spatial information of

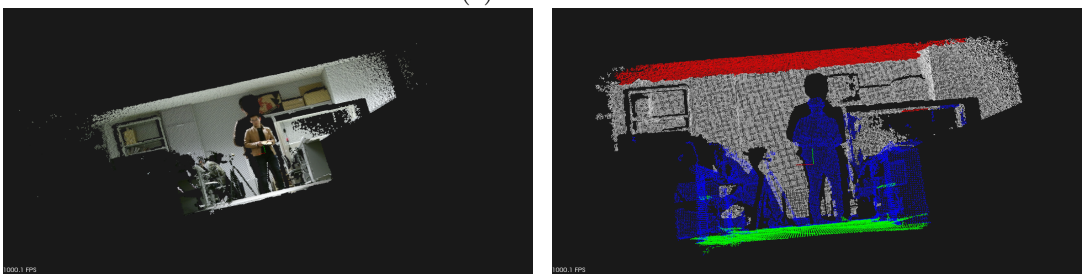


(a) frame 55



(b) frame 61

(c) frame 67



(d) frame 75

Figure 4.9: Labeling result of dataset3 (Part 2)

objects explicitly, which is less ambiguous than our feature vector. However, for the dataset with rotations, since position feature is not reliable anymore, unlike ours, which is more robust, this feature vector caused a failure result.

Since our labeling framework is proposed based on the work [69], we used this work

Table 4.4: Labeling and average accuracy for our dataset 1 with different features. Top half shows the results with no rotations, and the lower half shows the results by given an arbitrary rotation.

Method (Without rotation)	Roof	Wall	Candidates	Ground	Average
Position feature	98%	92%	90%	90%	92%
our work	95%	90%	87%	89%	90%
Method (With rotation)	**Roof**	**Wall**	**Candidates**	**Ground**	**Average**
Position feature	75%	71%	52%	70%	67%
our work	90%	85%	84%	85%	86%

Table 4.5: Labeling and average accuracy for our whole datasets.

Method	Roof	Wall	Candidates	Ground	Average
M.Bansal's work	75%	65%	60%	63%	66%
Daniel Wolf's work (RF)	80%	75%	78%	73%	76%
Daniel Wolf's work	84%	80%	79%	78%	80%
Our work(RF)	87%	85%	85%	86%	86%
Our work(without 2nd-potential)	93%	92%	87%	91%	91%
Our work	93%	93%	90%	91%	92%

for comparison. We also compared with M.Bansal's work [83]. First, labeling results obtained from a single RF classifer trained by our feature vector and the one used in [69] were compared, then we gave the results after using the CRF model without the second-smoothness potential. Finally we evaluated the performance of the proposed CRF model.

Figure 4.10 shows the comparable labeling results of different methods for each dataset and the accuracy is shown in Table 4.5. Comparison with the RF classification results by using the feature in [69] confirms that the proposed feature vector is robust to camera rotation and discriminative. However, as shown in the second row, many voxels were mislabeled by using RF classifier, which was caused by the independency. Using the CRF model in [69] refined some of them, but for the voxels having different surface but belonging to same label, this model is ineffective (third row). Since our CRF model was added by a second-smoothness potential, which can smooth the noise for irregular objects, our work achieved better results (fourth row) than [69].

For the person detection, we compared our feature vector with VFH to evaluate the performance. Detection results are shown in Fig.4.11. The performance of VFH is very effective and the average accuracy achieved 92%, while our feature vector improved the accuracy by 2%.

Our labeling and detection framework cost less than 800ms per frame such that it is usefull for 3D real time applications.

4.5 Conclusion

In this chapter, we have developed a set of robust feature vectors based on normal-deviation which are insensitive to camera rotation. Furthermore, by adding a new smoothness potential, we also defined a novel Conditional Random Field model to improve the performance. By using these feature vectors, we have proposed a framework for 3D indoor scene labeling with our CRF model and person detection. In additional to that, an approach for learning the contextual relations used for CRF has been proposed. Numerous experiments have been implemented and the results confirmed that our feature vectors and framework are robust to camera rotation.

For further research, we will focus on improving the calculation speed and developing a much stronger feature vector. We plan to use our framework for Pedestrian

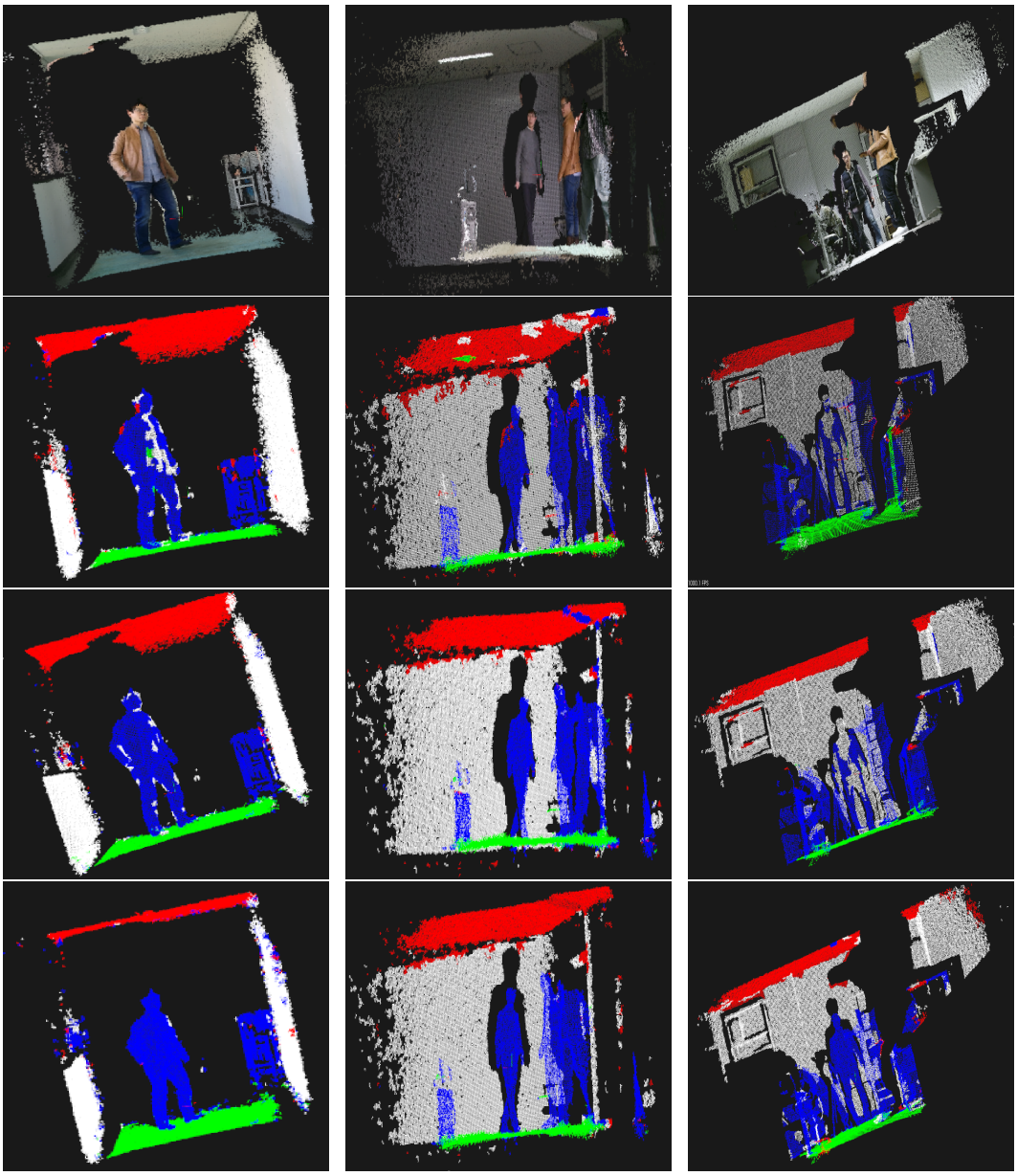


Figure 4.10: Experimental results. From the top to bottom: Input point cloud, RF results, general Dense CRF, and our CRF. Color used for the label: Red: Roof. White: Wall. Blue: Person Candidates. Green: Ground.

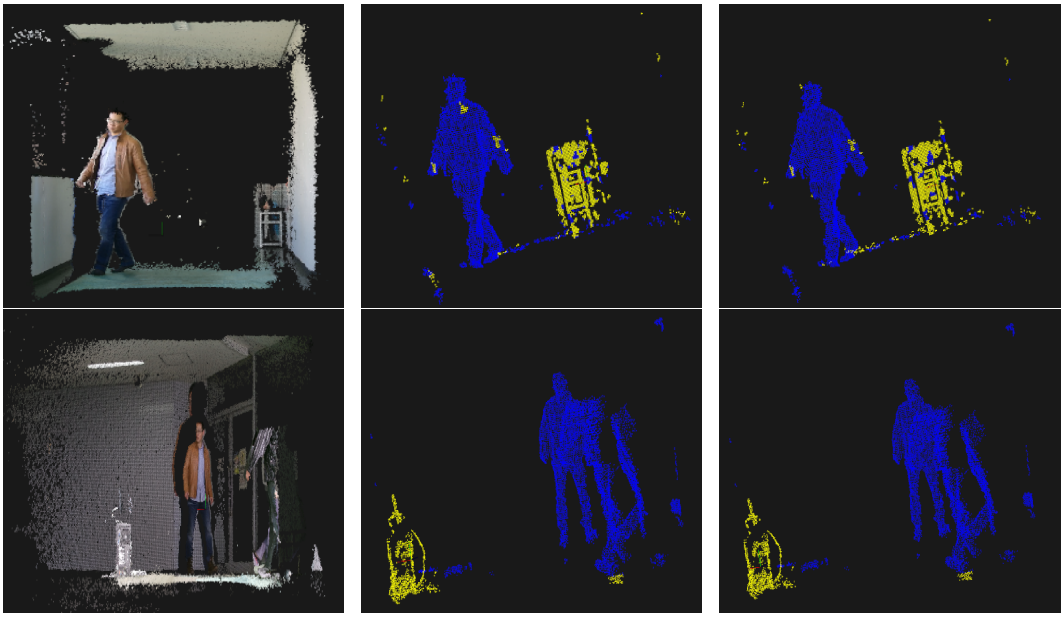


Figure 4.11: Performance of person detection with VFH (middle) and the feature we proposed (right)

tracking and detection since it largely reduced the number of the samples required to be determined.

5

Discussion and Future Work

5.1 Conclusion

In this thesis, we have proposed an algorithm for detecting single or multiple ground planes from a 3D point cloud, then given two Basyesian Graphical model-based algorithms for labeling 3D indoor scene.

Chapter 2 has given a framework that detects single or multiple ground planes with the estimation of the camera tilt angle. The contributions of this algorithm include:

1. In this algorithm, a method call θ-projection has been proposed. We have used this method to select the points belonging to the ground plane with a parameter, which is represented as the camera tilt angle. Another method of estimating the camera tilt angle from the selected points has been proposed. We have used these two methods iteratively to refine the final ground plane and the camera angle.

2. We have given a modification of the previous algorithm in 1 to improve the accuracy and running speed.

3. Based on the algorithm in 2, we have made an improvement to make it available to detect both parallel and non-parallel ground planes.

4. We have compared our algorithm with other state-of-the-art works to evaluate the performance.

Chapter 3 has given a Bayesian framework based on Markov Random Field that labels the 3D stationary indoor scene. The contributions of this algorithm include:

1. We have proposed a method to learn a novel spatial feature vector that derives from the combination of three oriental distribution by using eigenvector decomposition and sub-space combination.

2. We have designed a 6-connected model that describes the compatibility of label relationship

3. The final result is suffered from the uncertainty brought by the size of cube. To solve this problem, we have proposed a post precessing with the method proposed in chapter 2.

Chapter 4 has given another Bayesian framework that labels the 3D indoor scene with camera rotation. The contributions of this algorithm include:

1. To solve the problems of camera rotation, we have developed a set of feature vectors which are discriminative and robust to camera rotating based on the distribution of surface normal vectors.

2. To overcome the defects brought by MRF, we choose to use Conditional Random Field to solve the labeling problem. Meanwhile, we have proposed a method to learn the label relations rather than defining them manually, for the purpose of calculating the pairwise potential of the CRF.

3. For improving the performance, we have defined a new pairwise potential of the CRF.

4. We have developed another feature vector for detecting "person" objects from the labeling result.

5.2 Problems and future work

For ground plane detection, first, the range for selecting the ground points was set manually, which is not applied for all the cases. According to this, we aim to develop a learning method to find an appropriate range for each case. Second, since the location of the ground plane is indicated by the peak of the height distribution, it is necessary to develop a method to find the peak more efficiently in the aims of improving the performance and reducing the computational cost. Third, in this algorithm, only tilt

angle of the camera was considered. In the further research, we will take into account the roll angle to make this algorithm more applicable.

For 3D indoor scene labeling, first, manually defining or combining a set of features is not strong enough for every scene. For this reasaon, we aim to use convolutional neural network to learn the features. Second, for improving the performance, we will focus on learning the label relationships by using recurrence neural network. Finally, we aim to design a deep and strong architecture of the neural network for pixel-wise or point-wise segmentation.

References

[1] Mayank Bansal, Sang-Hack Jung, Bogdan Matei, Jayan Eledath, and Harpreet S Sawhney. **A real-time pedestrian detection system based on structure and appearance classification.** In *ICRA*, **10**, pages 903–909, 2010. 1

[2] Bastian Leibe, Nico Cornelis, Kurt Cornelis, and Luc Van Gool. **Dynamic 3d scene analysis from a moving vehicle**. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 1

[3] Dennis Mitzel and Bastian Leibe. **Taking mobile multi-object tracking to the next level: People, unknown objects, and carried items**. In *European Conference on Computer Vision*, pages 566–579. Springer, 2012. 1

[4] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. **Object detection with discriminatively trained part-based models**. *IEEE transactions on pattern analysis and machine intelligence*, **32**(9):1627–1645, 2010. 1

[5] Kirill Klionovski. **Theoretical and experimental research of diffraction on round semitransparent ground plane**. *IEEE Transactions on Antennas and Propagation*, **61**(6):3207–3215, 2013. 1

[6] Andreas Geiger, Julius Ziegler, and Christoph Stiller. **Stereoscan: Dense 3d reconstruction in real-time**. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968. IEEE, 2011. 2

[7] Chang Yuan and Gérard Medioni. **3D reconstruction of background and objects moving on ground plane viewed from a moving camera**. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, **2**, pages 2261–2268. IEEE, 2006. 2

[8] Daniel Maier and Maren Bennewitz. **Appearance-based traversability classification in monocular images using iterative ground plane estimation**. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4360–4366. IEEE, 2012. 2

[9] Anoop Cherian, Vassilios Morellas, and Nikolaos Papanikolopoulos. **Accurate 3D ground plane estimation from a single image**. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2243–2249. IEEE, 2009. 2, 10

[10] Lubor Ladickỳ, Paul Sturgess, Chris Russell, Sunando Sengupta, Yalin Bastanlar, William Clocksin, and Philip HS Torr. **Joint optimization for object class segmentation and dense stereo reconstruction**. *International Journal of Computer Vision*, **100**(2):122–133, 2012. 2

[11] Martin A Fischler and Robert C Bolles. **Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography**. *Communications of the ACM*, **24**(6):381–395, 1981. 2, 10

REFERENCES

[12] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. **RANSAC based out-of-core point-cloud shape detection for city-modeling**. *Proceedings of Terrestrisches Laserscanning*, 2007. 2

[13] Philip HS Torr and Andrew Zisserman. **MLESAC: A new robust estimator with application to estimating image geometry**. *Computer Vision and Image Understanding*, **78**(1):138–156, 2000. 3

[14] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. **Efficient RANSAC for point-cloud shape detection**. In *Computer graphics forum*, **26**, pages 214–226. Wiley Online Library, 2007. 3, 10

[15] `http://www.csse.uwa.edu.au/~pk/research/matlabfns`. 3

[16] Marco Zuliani, Charles S Kenney, and BS Manjunath. **The multiransac algorithm and its application to detect planar homographies**. In *IEEE International Conference on Image Processing 2005*, **3**, pages III–153. IEEE, 2005. 3, 10

[17] Roberto Toldo and Andrea Fusiello. **Robust multiple structures estimation with j-linkage**. In *European conference on computer vision*, pages 537–547. Springer, 2008. 3, 10

[18] Orazio Gallo, Roberto Manduchi, and Abbas Rafii. **CC-RANSAC: Fitting planes in the presence of multiple surfaces in range data**. *Pattern Recognition Letters*, **32**(3):403–410, 2011. 4, 10, 41, 107

[19] Xiangfei Qian and Cang Ye. **NCC-RANSAC: a fast plane extraction method for 3-D range data segmentation**. *IEEE transactions on cybernetics*, **44**(12):2771–2783, 2014. 4, 10, 107

[20] Faisal Mufti, Robert Mahony, and Jochen Heinzmann. **Robust estimation of planar surfaces using spatio-temporal RANSAC for applications in autonomous vehicle navigation**. *Robotics and Autonomous Systems*, **60**(1):16–28, 2012. 4, 40

[21] Dana H Ballard. **Generalizing the Hough transform to detect arbitrary shapes**. *Pattern recognition*, **13**(2):111–122, 1981. 5

[22] John Illingworth and Josef Kittler. **A survey of the Hough transform**. *Computer vision, graphics, and image processing*, **44**(1):87–116, 1988. 5

[23] Richard O Duda and Peter E Hart. **Use of the Hough transformation to detect lines and curves in pictures**. *Communications of the ACM*, **15**(1):11–15, 1972. 5

[24] Leandro AF Fernandes and Manuel M Oliveira. **Real-time line detection through an improved Hough transform voting scheme**. *Pattern Recognition*, **41**(1):299–314, 2008. 5

[25] Leandro AF Fernandes and Manuel M Oliveira. **Real-time line detection through an improved Hough transform voting scheme**. *Pattern Recognition*, **41**(1):299–314, 2008. 5

[26] Tahir Rabbani and Frank Van Den Heuvel. **Efficient hough transform for automatic detection of cylinders in point clouds**. *ISPRS WG III/3, III/4*, **3**:60–65, 2005. 5

[27] Nahum Kiryati, Yuval Eldar, and Alfred M Bruckstein. **A probabilistic Hough transform**. *Pattern recognition*, **24**(4):303–316, 1991. 5

[28] Jiri Matas, Charles Galambos, and Josef Kittler. **Robust detection of lines using the progressive probabilistic hough transform**. *Computer Vision and Image Understanding*, **78**(1):119–137, 2000. 5

[29] Jiri Matas, Charles Galambos, Josef Kittler, et al. **Progressive probabilistic hough transform**. 1998. 5

[30] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nüchter. **The 3D Hough Transform for plane detection in point clouds: A review and a new accumulator design**. *3D Research*, **2**(2):1–13, 2011. 6, 41

[31] Pekka Kultanen, Lei Xu, and Erkki Oja. **Randomized hough transform (rht)**. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, **1**, pages 631–635. IEEE, 1990. 6

[32] Jan W Weingarten, Gabriel Gruener, and Roland Siegwart. **Probabilistic plane fitting in 3D and an application to robotic mapping**. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, **1**, pages 927–932. IEEE, 2004. 7

[33] Lin Wang, Jianfu Cao, and Chongzhao Han. **Multidimensional particle swarm optimization-based unsupervised planar segmentation algorithm of unorganized point clouds**. *Pattern Recognition*, **45**(11):4034–4043, 2012. 7, 8, 107

[34] Junhao Xiao, Jianhua Zhang, Jianwei Zhang, Houxiang Zhang, and Hans Petter Hildre. **Fast plane detection for SLAM from noisy range images in both structured and unstructured environments**. In *2011 IEEE International Conference on Mechatronics and Automation*, pages 1768–1773. IEEE, 2011. 7

[35] GuruPrasad M Hegde and Cang Ye. **Extraction of planar features from swissranger sr-3000 range images by a clustering method using normalized cuts**. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4034–4039. IEEE, 2009. 8

[36] Dirk Hähnel, Wolfram Burgard, and Sebastian Thrun. **Learning compact 3D models of indoor and outdoor environments with a mobile robot**. *Robotics and Autonomous Systems*, **44**(1):15–27, 2003. 8

[37] Jann Poppinga, Narunas Vaskevicius, Andreas Birk, and Kaustubh Pathak. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3378–3383. IEEE, 2008. 9

[38] Tahir Rabbani, Frank Van Den Heuvel, and George Vosselmann. **Segmentation of point clouds using smoothness constraint**. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, **36**(5):248–253, 2006. 9

[39] Ahad Harati, Stefan Gächter, and Roland Siegwart. **Fast range image segmentation for indoor 3D-SLAM**. *IFAC Proceedings Volumes*, **40**(15):475–480, 2007. 9

[40] Kristiyan Georgiev, Ross T Creed, and Rolf Lakaemper. **Fast plane extraction in 3D range data based on line segments**. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3808–3815. IEEE, 2011. 9

[41] Junhao Xiao, Jianhua Zhang, Benjamin Adler, Houxiang Zhang, and Jianwei Zhang. **Three-dimensional point cloud plane segmentation in both structured and unstructured environments**. *Robotics and Autonomous Systems*, **61**(12):1641–1652, 2013. 9

[42] Navneet Dalal and Bill Triggs. **Histograms of oriented gradients for human detection**. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, **1**, pages 886–893. IEEE, 2005. 11

[43] Peter Gehler and Sebastian Nowozin. **On feature combination for multiclass object classification**. In *2009 IEEE 12th International Conference on Computer Vision*, pages 221–228. IEEE, 2009. 11

REFERENCES

[44] Jens Behley, Volker Steinhage, and Armin B Cremers. **Performance of histogram descriptors for the classification of 3d laser range data in urban environments**. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4391–4398. IEEE, 2012. 12

[45] Mayank Bansal, Bogdan Matei, Harpreet Sawhney, Sang-Hack Jung, and Jayan Eledath. **Pedestrian detection with depth-guided structure labeling**. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 31–38. IEEE, 2009. 12

[46] Georg Arbeiter, Steffen Fuchs, Richard Bormann, Jan Fischer, and Alexander Verl. **Evaluation of 3d feature descriptors for classification of surface geometries in point clouds**. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1644–1650. IEEE, 2012. 12, 14

[47] Bastian Leibe, Nico Cornelis, Kurt Cornelis, and Luc Van Gool. **Dynamic 3d scene analysis from a moving vehicle**. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 12

[48] Asma Azim and Olivier Aycard. **Layer-based supervised classification of moving objects in outdoor dynamic environment using 3D laser scanner**. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 1408–1414. IEEE, 2014. 12

[49] Asma Azim and Olivier Aycard. **Detection, classification and tracking of moving objects in a 3D environment**. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 802–807. IEEE, 2012. 12

[50] Andre Mouton, Toby P Breckon, Greg T Flitton, and Najla Megherbi. **3D object classification in baggage computed tomography imagery using randomised clustering forests**. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 5202–5206. IEEE, 2014. 12

[51] Jonathan T Barron, Mark D Biggin, Pablo Arbelaez, David W Knowles, Soile VE Keranen, and Jitendra Malik. **Volumetric semantic segmentation using pyramid context features**. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3448–3455, 2013. 12, 14

[52] Jens Behley, Volker Steinhage, and Armin B Cremers. **Performance of histogram descriptors for the classification of 3d laser range data in urban environments**. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4391–4398. IEEE, 2012. 12

[53] Rudolph Triebel, Kristian Kersting, and Wolfram Burgard. **Robust 3D scan point classification using associative Markov networks**. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2603–2608. IEEE, 2006. 12

[54] Andrew E. Johnson and Martial Hebert. **Using spin images for efficient object recognition in cluttered 3D scenes**. *IEEE Transactions on pattern analysis and machine intelligence*, **21**(5):433–449, 1999. 12

[55] Dragomir Anguelov, B Taskarf, Vassil Chatalbashev, Daphne Koller, Dinkar Gupta, Geremy Heitz, and Andrew Ng. **Discriminative learning of markov random fields for segmentation of 3d scan data**. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, **2**, pages 169–176. IEEE, 2005. 12

[56] Federico Tombari, Samuele Salti, and Luigi Di Stefano. **Unique signatures of histograms for local surface description**. In *European conference on computer vision*, pages 356–369. Springer, 2010. 12

[57] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. **Rotation invariant spherical harmonic representation of 3 d shape descriptors**. In *Symposium on geometry processing*, **6**, pages 156–164, 2003. 12

[58] Timothy Gatzke, Cindy Grimm, Michael Garland, and Steve Zelinka. **Curvature maps for local shape comparison**. In *International Conference on Shape Modeling and Applications 2005 (SMI'05)*, pages 244–253. IEEE, 2005. 12

[59] Mirela Ben-Chen and Craig Gotsman. **Characterizing Shape Using Conformal Factors.** In *3DOR*, pages 1–8, 2008. 12

[60] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. **Aligning point cloud views using persistent feature histograms**. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391. IEEE, 2008. 12

[61] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. **Fast point feature histograms (FPFH) for 3D registration**. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009. 13, 76, 77

[62] Berthold Klaus Paul Horn. **Extended gaussian images**. *Proceedings of the IEEE*, **72**(12):1671–1686, 1984. 13

[63] Richard J Campbell and Patrick J Flynn. **Eigenshapes for 3D object recognition in range data**. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, **2**. IEEE, 1999. 13

[64] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. **Shape distributions**. *ACM Transactions on Graphics (TOG)*, **21**(4):807–832, 2002. 13

[65] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. **Fast 3d recognition and pose using the viewpoint feature histogram**. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162. IEEE, 2010. 13, 82, 83

[66] Hema S Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. **Semantic labeling of 3d point clouds for indoor scenes**. In *Advances in Neural Information Processing Systems*, pages 244–252, 2011. 14

[67] Mohammad Najafi, Sarah Taghavi Namin, Mathieu Salzmann, and Lars Petersson. **Non-associative higher-order markov networks for point cloud classification**. In *European Conference on Computer Vision*, pages 500–515. Springer, 2014. 14

[68] Mayank Bansal, Bogdan Matei, Harpreet Sawhney, Sang-Hack Jung, and Jayan Eledath. **Pedestrian detection with depth-guided structure labeling**. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 31–38. IEEE, 2009. 15, 62, 67, 107

[69] Daniel Wolf, Johann Prankl, and Markus Vincze. **Fast semantic segmentation of 3D point clouds using a dense CRF with learned parameters**. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4867–4873. IEEE, 2015. 15, 16, 79, 90, 92, 107

[70] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. **Indoor segmentation and support inference from RGBD images**. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012. 15

[71] Xiaofeng Ren, Liefeng Bo, and Dieter Fox. **Rgb-(d) scene labeling: Features and algorithms**. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2759–2766. IEEE, 2012. 16

[72] Julien PC Valentin, Sunando Sengupta, Jonathan Warrell, Ali Shahrokni, and Philip HS Torr. **Mesh based semantic modelling for indoor and outdoor scenes**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2067–2074, 2013. 16

REFERENCES

[73] Alexander Hermans, Georgios Floros, and Bastian Leibe. **Dense 3d semantic mapping of indoor scenes from rgb-d images**. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2631–2638. IEEE, 2014. 16, 80

[74] Abhishek Anand, Hema Swetha Koppula, Thorsten Joachims, and Ashutosh Saxena. **Contextually guided semantic labeling and search for three-dimensional point clouds**. *The International Journal of Robotics Research*, page 0278364912461538, 2012. 16

[75] Olaf Kahler and Ian Reid. **Efficient 3d scene labeling using fields of trees**. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3064–3071, 2013. 17

[76] Jeremy Jancsary, Sebastian Nowozin, Toby Sharp, and Carsten Rother. **Regression Tree FieldsAn efficient, non-parametric approach to image labeling problems**. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2376–2383, 2012. 17

[77] Byung-Soo Kim, Pushmeet Kohli, and Silvio Savarese. **3D scene understanding by Voxel-CRF**. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1425–1432, 2013. 17

[78] Omid Hosseini Jafari, Dennis Mitzel, and Bastian Leibe. **Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras**. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5636–5643. IEEE, 2014. 22

[79] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 27, 58

[80] Alexey Abramov Jeremie Papon and Markus Schoeler. **Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds**. *CVPR*, pages 2027 – 2034, 2013. 76

[81] P. Krahenbuuhl and V. Koltun. **Parameter Learning and Convergent Inference for Dense Random Fields**. *ICML*, 2013. 82

[82] Solomon Kullback and Richard A Leibler. **On information and sufficiency**. *The annals of mathematical statistics*, **22**(1):79–86, 1951. 82

[83] H. Sawhney S. Jung M. Bansal, B. Matei and J. Eledath. **PedestrianDetection with Depth-guided Structure Labeling,**. *ICCV Workshops*, pages 31–38, 2009. 92

List of Figures

List of Tables